Rule based phonetic search approaches for central Europe

Conference Paper · October 2010

DOI: 10.1109/SISY.2010.5647210 · Source: IEEE Xplore

Ivan Polášek
Comenius University Bratislava
30 PUBLICATIONS **206** CITATIONS

Text Search of Surnames in Some Slavic and Other Morphologically Rich Languages Using Rule Based Phonetic Algorithms

Dušan Zahoranský and Ivan Polasek

***Abstract*—Surnames play a key role as person natural identifiers, essentially in present information systems. This paper deals with the topic of optimizing a phonetic search algorithm as a string matching of surnames usable for communications service providers, person registries, social networks or genealogy databases. It describes a proposed solution for the phonetic searching of Slovak and (territorial) neighboring languages (Czech, Polish, Ukrainian, Russian, German, Hungarian, Jewish) surnames. This solution was designed to improve search precision and recall when searching for people by their surnames originating in these languages.**

***Index Terms*—Algorithms, information retrieval, natural languages.**

I. Introduction

FINDING information related to an imperfectly identified surname may be so important, that it is worth researching special methods for conducting such a search. For example, let us consider emergency calls, where identifying a person may be of vital importance. However, the surname may be given imprecisely, e.g. because the caller is not sure of the surname or exact spelling of the surname or due to errors introduced by the operator while making the transcript to text for a search query input.

Algorithms aimed to solve these problems exist, but no algorithm can completely solve all types of errors or inaccuracies. Individual algorithms primarily differ in the target language group to which they are intended for, or the typographical errors which they are able to recognize.

The phonetic matching approach can reduce input query errors based on the phonetic pronunciation of a word in a specific language. The phonetic search tries to match strings that have a similar pronunciation, regardless of the actual spelling. A typical application is a “white pages” enquiry line, where a telephone operator is verbally given a name, guesses the spelling (or is provided with a spelling, which may be incorrect) and uses the guess, to query a database of surnames. The phonetic matching system should then find a misspelled name in the database [26]. Another present time example might be a search for people on a social network or genealogical database, when the user does not know the spelling of the surname, just the pronunciation and tries to figure out the actual spelling.

The topic of research reported in this paper is not speech recognition. The paper deals with a phonetic matching of surnames in text form. A search query is a surname in the form of a string, which is used to undertake a search against surnames stored in the data repository. The initial idea for the research was instigated in recent years through a demand from the person data retrieval system, which is capable of identifying the person from an inexact or misspelled surname in a query. The algorithm is currently prepared for industrial usage in real-time retrieval systems.

Research does not cover recognition of the surnames in free text. A search algorithm intention is to eliminate possible typing errors in the search query caused by similar pronunciation of some letters or syllables in the surnames and provides a very effective way of matching surnames using indexed canonical forms for the stored surnames.

Most of the existing phonetic algorithms are intended just for the English language or partially support other languages, but they are not primary intended for the Slavic or territorially close languages, which are the target of our research. Usage of these algorithms for different languages therefore yields many irrelevant hits, furthermore missing many of relevant ones. Our research focus is on different middle and east-European languages with the goal to intend an algorithm suitable for eliminating phonetic errors when searching for common surnames of people living in Slovakia and territorially close countries. The proposed algorithm especially focuses on the phonetics of the Slovak language–how Slovak surnames might be misspelled, but also covers many rules for other middle European languages. Rules are new or based on other existing rule-based phonetic algorithms.

This paper also presents a comparison of some well-known phonetic algorithms in case of searching for surnames in languages targeted by our research. By transliterating an input string we were able to perform a search with these algorithms and measure their result using some common metrics.

We published a first proposal of our approach in [25] and here we present the algorithm with a complete set of the rules and evaluation.

Manuscript received May 21, 2013; revised February 28, 2014; accepted September 24, 2014. Date of publication January 16, 2015; date of current version February 26, 2015. This work was supported in part by the Scientific Grant Agency of Slovak Republic under Grant VG 1/1221/12. It is a partial result of the Research and Development Operational Programme for the project Research of methods for acquisition, analysis, and personalized conveying of information and knowledge, ITMS 26240220039, co-funded by the ERDF. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Thomas Hain.

The authors are with Software Engineering, Faculty of Informatics and Information Technology, Slovak University of Technology (FIIT STU), 821 04 Bratislava, Slovakia (e-mail: dusanzahoransky@gmail.com; ivan.polasek@stuba.sk).

Digital Object Identifier 10.1109/TASLP.2015.2393393

The rest of the paper is structured as follows. In Section II we discuss some existing search approaches. More can be found in [25]. Section III describes known phonetic algorithms. Sections IV, V and VI present our approach for the phonetic matching of Slavic surnames. Section VII shows the evaluation of our approach. Conclusions and suggestions for future work are discussed in Section VIII.

II. Existing Search Approaches

The first-hand and elementary approach is the exact match, when records in the search results must exactly match the input query. The second one is a partial or a pattern match, when a query contains an uncertainty, which may be masked with wildcard characters, or a query is only modelled to match the expected records e.g. match using regular expressions. Another approach is called plain string similarity (distribution is based on [8][14]), which is based on the comparison of similar groups of letters in two words not regarding their position. Another approach is to try to eliminate typing errors. It calculates similarity as a minimum count of transformations required to transform the first word to the second one. There are also grammatical algorithms, including stemming and synonym matching, which can be used for Slovak language as well [17].

The last approach is phonetic similarity, represented by language dependent string matching algorithms. Hybrid matching algorithms compose more algorithms and approaches [20]. *Personal Name Recognizing Strategy* (PNRS) [22] is a language identifier and a decision mechanism, combining a *Restricted Near Miss Strategy* (RNMS) where two records are *near*, if they are identical after inserting a blank space, interchanging adjacent letters, swapping, changing, deleting or inserting one letter and *SoundD* phonetic strategy (modified *Soundex* algorithm with initial letter parsing rules and reducing result code to 4 digits number). The following subsections describe some of the well-known representatives of the different search approaches, with a focus on phonetic algorithms, which are related to the goal of our research.

A. Algorithm of plain string similarity

The N-grams algorithm (in some literature referred to as Q-grams [5]) is the representative of the plain string similarity. A search using n-grams is considered to be independent of the language. N-grams are n-character sequences where each sequence begins with the following character of the word. The process of the two strings comparison follows these steps:

1) Create a set of n-grams for each word.
2) Create an intersection of n-grams sets.
3) The more n-grams this intersection contains, the more similar are the words

A coefficient of similarity S, of two n-grams is defined by equation (1), where N_1 is a set of n-grams of the first word and N_2 are n-g-rams of the second one.

$$S = \frac{|N_1 \cap N_2|}{|N_1 \cup N_2|} \qquad (1)$$

B. Elimination of Typing Errors

The Damerau-Levenshtein metric focuses on the strings similarity with regard to errors that can arise when writing a word. These errors can be divided into four groups:

1) The insertion of an additional letter.
2) The deletion of a letter.
3) The substitution of one or more letters.
4) The transposition of adjacent letters.

The distance of two words can be computed by creating a matrix, where the first row is filled with the letters of the first word and the first column with the letters of the second one. Then the matrix is filled from the top left corner, using the recursive formula $edit(i, j)$, where i and j are the length of the words and $edit(i, j)$ is the restricted Damerau-Levenshtein edit-distance of the substrings of these words (transformation one substring to the second one).

C. Phonetic approach

Phonetic algorithms are intended to eliminate search query errors caused by phonetic pronunciation of the word in different languages. Most phonetic algorithms transform a word into a canonical form. The canonical form is a string, produced by the application of the transformation rules defined by the specific algorithm to the original word. All words having the same canonical form as the query word will be matched [13].

Most of the phonetic algorithms are intended just for the English language, for instance Soundex, NYSIIS, Phonix or Metaphone (see the Section C.1–C.4). But there are also algorithms which have been adapted to Slavic phonetics, such as Double Metaphone and Daitch-Mokotoff Soundex (see the Sections II-C.5 and II-C.6). In our research we also adapted and tested algorithms intended for English surnames.

Most of the well-known phonetic algorithms are rule based. They define a set of transformation rules. Rules are defined for a classical Latin alphabet–character set from A to Z. Many middle Europeans languages however consist also of diacritical characters e.g. í,č,ô,ä,... To adapt these algorithms we had to use a transliteration of diacritical letters of the input query string.

According to the way the data is stored in the repository two search techniques can be applied, namely on-line and off-line searching. The off-line search approach algorithm uses preprocessed data in the repository. When a surname is inserted into a repository, the canonical form is computed and stored in the repository too. Usually an index is created on these canonical forms to achieve better speed during the search. The on-line search approach generates canonical forms of words during the search process. The advantage of the on-line search is that there is no need for preprocessing when inserting data into the database, which results in less space and time needed to store the records. The disadvantage is a considerably slower response, which may cause a problem, especially with larger quantities of data. Some of the algorithms do not support off-line search. They do not just match the canonical form, but they also need to perform some additional comparisons of each surname with the surname from the search query.

Rule based phonetic algorithms use a set of transformation rules which reflect the phonetics of a specific language or languages. To provide a phonetic algorithm for multiple languages

TABLE I
PHONETIC CODES FOR SOUNDEX [26]

Code	0	1	2	3	4	5	6
Letters	aeiouy hw	bp fv	cgjkq sxz	dt	l	mn	r

TABLE II
PHONIX PHONETIC CODES [26]

Code	0	1	2	3	4	5	6	7	8
Letters	aeiouy hw	bp	cgjkq	dt	l	mn	r	fv	sxz

we need to adapt more rules. More rules cause more surnames to be transformed to the same canonical form. A difficult task in algorithm design is to try to define the rules in such a way that only surnames, considered phonetically the same, will be transformed to the same canonical form. In general the greater the number of transformation rules, the larger is the hitlist retrieved by the algorithm.

Evaluation of phonetic algorithms is another difficult task. A flaw is the absence of precise metrics to assess the similarity of two surnames with each other [6]. Everyone could have a different opinion about the relevance of a match returned by a phonetic algorithm.

In the following subsections we give a description of the most common phonetic algorithms, including a short description of the search principles, target languages and their advantages and disadvantages.

Soundex: The Soundex algorithm [12] has become the most widely used phonetic algorithm, although the disadvantage is that it is built only for the English language. Usage of Soundex in different languages yields larger amounts of not relevant hits, which yields poor precision [20][2]. Soundex uses codes based on the pronunciation of each character in a word [26][6].

The algorithm uses these four rules to convert words into canonical form.

1) Replace all the letters except the first one with a phonetic code (Table I).
2) Remove all repeated adjacent codes (collapse adjacent similar digits into one).
3) Remove all codes "0" (the codes representing vowels and letters H and W).
4) Return the first four characters of the transformed string (if a string is too short, use padding with zeros).

Phonix: The Phonix algorithm was created in 1988 as an improvement of the original Soundex algorithm [3], [4]. It uses the same algorithm as Soundex, but different codes (Table II). Like the Soundex algorithm the Phonix focuses on the English language. Furthermore, the transformation to the canonical form is enriched by many transformations.

NYSIIS: The "New York State Identification and Intelligence System" (NYSIIS) [21] was proposed for the criminal justice system in the New York State Division, where it is still used today (updated in July 2006). The algorithm is built for searching English sounding surnames. According to [1], the difference between the Soundex and NYSIIS algorithm is that NYSIIS retains information about the positions of the vowels in a word. NYSIIS also replaces letters in transformation with other phonetically similar letters instead of numerical codes.

Metaphone: This algorithm is another variant of the original Soundex algorithm. Like the Soundex it is based on English phonetics. Metaphone was created by Lawrence Phillips in 1990 [15]. It contains a richer set of rules and is better able to detect different-sounding names than the Soundex algorithm.

Double Metaphone: Double Metaphone was designed by Lawrence Phillips about ten years later in 2000 as a descendant of the Metaphone algorithm [16]. As stated [18], the Double Metaphone algorithm tries to adapt to the phonetics of languages other than of English (Slavic, Germanic, Celtic, French, Italian, Spanish and even Chinese). It also has more rules and greater complexity than its predecessor.

Daitch-Mokotoff Soundex: The algorithm was developed by Randy Daitch and Gary Mokotoff of the Jewish Genealogical Association as an improvement of the Soundex to adapt it also for Jewish and Slavic surnames [10]. The Daitch-Mokotoff Soundex contains a refinement to support multiple language surnames not just for English [18].

The Daitch-Mokotoff Soundex contains a rule for each alphabet letter. The rule may contain alternatives, which are the groups of characters starting with a character to which the rule is designed with its own phonetic codes. There is just one rule for each character. The transformation therefore can clearly identify the rule for a processing letter. The algorithm then looks for an alternative. The goal is to find the longest possible alternative, which starts from the currently processed letter and matches the following letters. If a suitable alternative is found, the algorithm uses the phonetic codes of the alternative, otherwise it uses the phonetic codes of the rule.

The rule contains several phonetic codes. The decision about which code is used is determined by the position of the letter in the word and by the following letters. The rule contains a phonetic code for the first letter of the word, letters followed by a vowel and for other cases.

Editex: The algorithm Editex [27] was created to measure the phonetic distance of two words. It combines the features of the Edit distance and the Soundex algorithm. The difference is that function $r(a, b)$, which assign the similarity of two letters can results in three values 0, 1 or 2. The $r(a, b)$ result is 0 if a and b are identical, one if a and b are located in the same group of characters (according to the distribution in Table III), and 2 otherwise. Another difference compared to the Edit distance is a new function $d(a, b)$, which is similar to function $r(a, b)$, with the difference that if one of the letters is "h" or "w" and at the same time "a" is not equal to "b," then $d(a, b)$ is equal to 1. The recursive definition of the algorithm is shown in (2).

$$\begin{aligned}
edit(0,0) &= 0 \\
edit(i,0) &= edit(i-1,0) + d(s_{i-1}, s_i) \\
edit(0,j) &= edit(0,j-1) + d(t_{j-1}, t_j) \\
edit(i,j) &= \min\,[edit(i-1,j) + d(s_{i-1}, s_i), \\
&\qquad edit(i,j-1) + d(t_{j-1}, t_j), \\
&\qquad edit(i-1,j-1) + r(s_i, t_j)]
\end{aligned} \tag{2}$$

TABLE III
EDITEX CODING RULES

Code	0	1	2	3	4	5	6	7	8	9
Letters	aeiouy	bp	ckq	dt	lr	mn	gj	fvp	sxz	csz

III. PHONETIC MATCHING OF SLAVIC SURNAMES

The Slovak language is a part of the Slavic language family. In general it is phonetically similar to other Slavic languages, but also very different from English. Mainly for historical reasons, Slovak surnames are similar to, or derived from, Slavic languages or surrounding states' surnames.

Slavic languages have some differences compared to English. These languages use diacritical characters. All well-known phonetic algorithms, which we had the opportunity to test, cannot transform diacritical characters. To adapt them for our tests we used transliteration, which maps diacritical letters to a basic alphabet letter.

Another difference is related to Slavic surnames. An important part of Slavic surnames is the ending syllable. In English surnames it plays almost no role. Phonetic algorithms often use a canonical form, which has a maximum length of 4 or a maximum length of 6 characters. It means the transformation of the original word into a canonical form ends as soon as the canonical forms reach the maximum character length. The rest of the word is simply omitted. It often happens that the canonical forms do not contain phonetic codes of trailing characters of the surname. However according to the final syllable of a Slavic surname it can be often decided whether the surname is female or male.

In an English surname the pronunciation of vowels is often silent, or dependent on the surrounding consonants. Also the pronunciation of different vowels may be similar. For example pronouncing the surnames "Keiller" and "Keilor." The vowels "e" and "o" in both given surnames are pronounced as "ə." A similar example of assonance is the surname "Knollys" and "Knowles" or "Rees" and "Rhys." In the surname "Rhys" moreover the letter "h" is silent. English phonetic algorithms therefore disregard vowels and the consonant H. In Slovak language the sound of these letters is accentual and therefore a phonetic algorithm should not ignore them.

A. Origin of Surnames on the Territory of Slovakia

On the territory of Slovakia, several languages have been used in the past, for example Latin as the cultural and sacred language, or Hungarian as the official language in Austria-Hungary. There are also close associations with languages such as German, Czech, Polish, Ukrainian or Russian. Mutual relations between Slovak and those languages were also created through the migration of inhabitants [11]. This natural multilingual context is important to take into account when creating a phonetic algorithm for Slovak surnames. This close interrelatedness of Slovakia with the surrounding languages meant that many Slovak surnames originated from foreign languages.

Because of this fact a phonetic algorithm for Slovak surnames must be suitable for searching surnames of the surrounding states. The major emphasis of the algorithm presented in the paper lies clearly on Slovak surnames but also Slovak surnames derived from surrounding countries e.g. Ukraine, Russia, Poland, Czech, Hungary and also surnames of Jewish and Germanic origin.

Most of the algorithms that were tested during the research were not adapted for searching in this wider region, except the algorithms of the Daitch-Mokotoff Soundex and Double Metaphone which take into account other phonetics other than English. They distinguish vowels and have certain rules for Slavic languages. Even these two algorithms, however, were not able to obtain good precision for matching surnames in our test collection.

B. Our First Approach: FuzzySK Layer

Algorithm FuzzySk is a transformation layer standing above a phonetic algorithm. The string transformed by FuzzySk layer is passed as an input query for a phonetic algorithm. Using this approach a word can be adapted to specific language phonemes without the need of the phonetic algorithm rules modification. Rules for different groups and target languages can be created and can be interchanged as is needed. The transformation process using FuzzySk is shown in Fig. 1. Unfortunately this approach did not bring the satisfactory results as the precision of the result was poor. Despite the presence of FuzzySk transformation, the underlying phonetics algorithms retrieve a lot of irrelevant hits, the surnames which were entirely different from the surname in the query according to Slavic languages phonemes.

IV. OUR MAIN APPROACH–RULES PRECONDITIONS AND RULES EXTENSION: MIDEPHONE

Our algorithm MidEPhone (Phonetic algorithm especially for Middle and East Europe) is proposed for surnames originating in Slovakia and the surrounding countries in the central and east European region or for Slavic and other morphologically rich languages (Jewish, German, Hungarian, etc.). The principles of the algorithm are explained in this section.

A. The Rule for Each Alphabet

The rule for each basic alphabet letter is defined. A rule may contain alternatives with the groups of letters starting with a letter to which the rule is designed. The alternatives have their own phonetic codes. The principle of rules and alternatives was inspired by the Daitch-Mokotoff Soundex algorithm.

The rule has several phonetic codes assigned to it. The decision about the code used in the transformation is determined by the position of the character in the word and the following letter. The rules contain codes for the first letter of the word, for a letter followed by a vowel, the last letter in the word (or rule ending alternatives - groups of letters at the end of word) and the phonetic code for other cases.

The canonical form does not include adjacent duplicate characters. If the last character in the canonical form is identical to the first character of the appending phonetic code, then only the substring after the first character of the phonetic code is appended into canonical form.

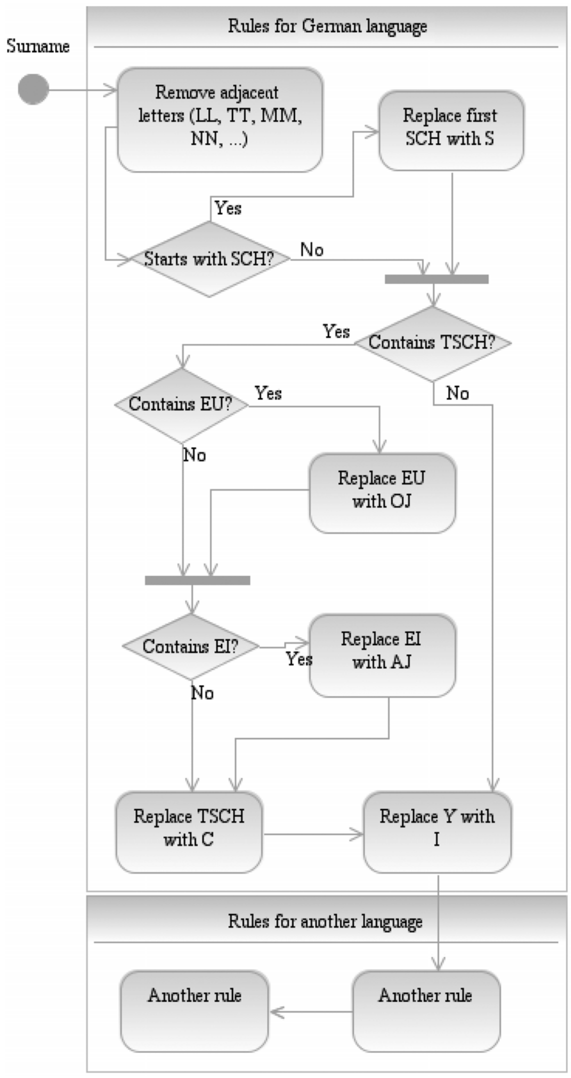

Fig. 1. Example of FuzzySk transformation rules.

TABLE IV
EXAMPLE OF SURNAMES TRANSFORMATION WITH MIDEPHONE ALGORITHM

Surname	Code	L*	Transformation								LS Tran**
Čajkovský	S3K1SK	6	Letters	C	A	J			KOV	S	KY
			Codes	S	3	3			K1	R	K
Tchaykovsky	S3K1SK	6	Letters	TCH	A	Y			KOV	S	KY
			Codes	S	3	3			K1	R	K
Petrovič	13BR213S	8	Letters	P	E	T	R	O	V	I	C
			Codes	1	3	B	R	2	1	3	S
Petrovics	13BR213S	8	Letters	P	E	T	R	O	V	I	CS
			Codes	1	2	L3	S	0	1	3	9
Halász	KL09	4	Letters	H		AL					ASZ
			Codes	K		L			0		9
Halaš	KL09	4	Letters	H		AL					AS
			Codes	K		L			0		9
Smidth	S43B	4	Letters	S		M			I		DTH
			Codes	S		4			3		B
Šmíd	S43B	4	Letters	S		M			I		D
			Codes	S		4			3		B
Alexandrov	LKSV	4	Letters	AL			X				V
			Codes	L			KS				V
Alexandrovský	LKSV	4	Letters	AL			X				VSKY
			Codes	L			KS				V
Alekseev	L3KB	4	Letters	AL		E			K		V
			Codes	L		3			3		B
Alexej	L3KB	4	Letters	AL		E			X		J
			Codes	L		3			K		B

*Length of the canonical form
**The last syllable transformation

B. The Canonical Form Length

We undertook experiments with the three different maximum lengths of canonical forms: four, six or eight characters. All, except the last character of canonical form, are produced by transformation of the input string, processed from the beginning of a word towards the end. The last character is created by the transformation of the last letter, or group of last letters when using alternatives.

Through the transformation the rule for a processing letter is clearly assigned. After the rule for the letter is chosen, the algorithm tries to find the longest applicable alternative of the rule. The applicable alternative means, the alternative has to match the substring which starts from the currently processing letter. If such an alternative is found, the algorithm uses the phonetic coding of this alternative, otherwise the phonetic codes of the current rule are used.

C. Short Form Padding

If the surname is too short and the canonical form does not reach the required length, it is filled up with zero characters. The algorithm appends the required number of "0" characters into the canonical form string between the characters obtained by the transformation of a word from the beginning and characters obtained by the transformation of the last character. For example, the result of the transformation of the surname "Schuster" into the six character long canonical form will be the string "S2S00R." The transformation of the last character creates the phonetic code "R," the transformation of the word from the beginning creates the code "S2S." To achieve the length of six characters, it will be necessary to insert two zeros between these two parts of the code.

D. The Last Syllable Transformation as an Essential Contribution

The innovation of our approach compared with the principle of the Daitch-Mokotoff Soundex algorithm is the introduction of the rules ending alternatives - transformation of the last letter or group of ending letters. This concept should bring a significant increase in precision, because they are often the key in identifying gender or the significant accent on pronunciation lies upon them.

The first step of the algorithm is the transformation of the last letters. The algorithm chooses the rule based on the last letter of the input word. If the rule contains applicable ending alternatives for the end of the word, it tries to find the longest one. The applicable alternative has to match the letters at the end of the word. The phonetic code is appended at the end of the canonical form after the transformation finish. The transformation then keeps processing the word from the beginning with the unprocessed part of the input string.

Table IV shows surnames encoded with the MidEPhone algorithm. The phonetic codes used during the transformation

TABLE V
RULES OF MIDEPHONE ALGORITHM

Rule			Phonetic code			
L	**RA**	**REA**	**FL**	**BW**	**LL**	**O**
A			1A	A	A	3
	AU, AUE, AYE		1A	A		A
	AL		L	L		L
		CA			C	
B			1	1	B	B
	BO, BE		B3	B3		B3
C			S	S	S	S
	CH		K	K		K
	CZ		S	S		S
		IAC, AC			9	
D			1	1	B	B
	DN		4	4		4
E			3	3		3
	EO, EU		2	2		2
	EL		L	L		L
		EE	E	E	R	E
F			1	1	F	F
G			K	K	G	K
	GOL, GUL		KL	KL		KL
H			K	K		K
		ECH, ICH, ACH, IACH, IECH, TH			9	
I			3	3	I	3
	IA		3	3		3
	IL		L	L		L
J			1	3	J	3
K			K	K	K	K
	KOV, KOP, KOT, KOB, KOD, KUV, KUP, KUT, KUB, KUD		K1	K1		K1
	KOR		KR	KR		KR
	KOL, KUL		KL	KL		KL
	KUR		KR2	KR2		KR2
		IK, EK, AK, IAK, IEK, JAK, JEK, JIK			9	
L			L	L	R	L
	LO, LA		L3	L3		L3
M			4	4	M	4
	MO, MA		M3	M3		M3
N			4	4	N	4
		EON, ON, AN, JON			N	
O			2	2	O	2
		CO			C	
P			1	1	B	B
R			R	R	R	R
	RIK		RK	RK		RK
	RNA		R34	R34		R34
	RIKOV, RIKOP, RIKOT, RIKOB, RIKOD, RIKUV, RIKUP, RIKUT, RIKUB, RIKUD		RK1	RK1		RK1
		TER, TR			R	
S			S	S	9	S
	SP		S	1		B
	SHT, SCHT, SCHD, ST, SZT, SHD, SZD, SD		SD	SD		SD
	SV, SL, SZL		SV	SV		SV
	SO, SA		S3	S3		S3
	SCH, SH, SZ, SS		S	S		S
		ZS, AS, IAS, AZS, IAZS			9	
T			1	1	B	B
	TH		B	B		B
		DT			B	
U			2	2	2	2
V			1	1	1	F
W			1	1	1	F
X			KS	KS	KS	KS
Y			3	3	I	3
	YL		L	L		L
		KY			K	
		VSKY			V	
Z			S	S	9	S
		SZ, IASZ, ASZ, TZ			9	

Columns headers: L-Letter, RA–Rule alternative, REA–Rule ending alternative, FL–First letter, BW–Before a vowel, LL–Last letter, O–Other cases.

process and also different lengths of the canonical form can be seen in the table as well.

E. *Rules of MidEPhone Transformation*

Table V contains a complete list of the rules for the algorithm coding. An empty cell in the table means that the character is not transformed to any phonetic code, thus nothing is appended into the canonical form.

F. *The Core Transformation Using MidEPhone*

Transformation into a canonical form of the maximum length of four characters, using the MidEphone algorithm is shown using the surname "Kollár." It consists of these steps.

1) Uppercase the string, result is "KOLLÁR."
2) Transliterate the string, result is "KOLLAR."
3) Perform the transformation of the last letter of the surname.
4) Last letter is "R"; therefore find the rule for letter "R."
5) Try to find ending alternatives of the rule. The rule contains alternatives "TER" and "TR," but "TER" does not match "LAR," as well as "TR" does not match "AR" at the end of the string "KOLLAR."
6) Continue with the transformation of the substring without the last letter processing it from the beginning towards the end. The substring is "KOLLA."
7) Store the phonetic code of the rule for the letter "R" in the column "Last letter," which is the character "R."
8) First letter is "K"; therefore find the rule for letter "K."
9) Try to find alternatives of the rule. The rule contains alternatives, one with the match that is "KOL."
10) Append into canonical form the phonetic code of the rule alternative "KOL" in the column "First letter," which is "KL."
11) Following the unprocessed letter is the second "L" in the string "KOLLA," therefore find the rule for the letter "L."
12) Try to find alternatives of the rule. The rule does not contain any alternatives.
13) Append into canonical form the phonetic code of the rule "L" in the column "Before a vowel" (because the next letter is vowel "A"), which is "L."
14) Last character in the actual canonical form is "L" therefore appending of "L" will not be performed.
15) Following the unprocessed letter is the second "A" in the string "KOLLA," therefore find the rule for letter "A."
16) Try to find alternatives of the rule. The rule contains alternatives "AU," "AL" and "AUE," but none match the letter "A" at end of the string "KOLLA."
17) Append into canonical form the phonetic code of rule "A" in the column "Other cases," which is "3."
18) The actual canonical form is the string "KL3," its length is 3, the last character appended is the stored phonetic code created by the transformation of the last letter, the character "R."
19) The final canonical form is the string "KL3R."

The algorithm then uses the canonical form "KL3R" of the surname "Kollár" to select surnames by matching their canonical forms in the data store.

G. *Simplified Code of the Transformation Process*

The code snippet on the following page, is a simplified version of the algorithm implementation, it shows the core steps in the transformation process.

Algorithm 1: Transformation process, the core steps.

```
Transformation algorithm:
    Input: Surname, Canonical form length
    Output: Canonical form
begin
    transform surname to upper case;
    transliterate surname;
    find a rule for the surname last letter;
    if (ending alternative exists)
        use "ending alternative code" as the last of the
        canonical form
    else use "rule last letter code" as the last of the
    canonical form;
    while (canonical form length is not reached and
    unprocessed letters exist)
        if (alternative exists) use alternative to fetch code
        else use rule to fetch code;
        if (processing first letter)
            append "first letter code" to the canonical form
        else if (vowel is next)
            append "before vowel code" to the canonical form
        else append "code" to the canonical form;
    append padding;
end
```

V. The Evaluation of MidEPhone

The evaluation of phonetic algorithms itself is a difficult task. There are a few techniques to measure the quality of phonetic algorithms; all of them need a test set of queries with an assessment of hits relevance, which requires a subjective judgment. For the creation of the test set the Relevance Judgment technique is typically used. Relevance judgments specify which documents from the datastore are relevant for a particular query. Human assessors judge the relevance of each document in the collection to all the queries. It assumes that if the information retrieval performs well on one set of judgments, it will perform well on others as well [24].

An inevitable part in the development and evaluating our phonetic algorithm is the large set of data, relevant to the languages, that we intend the algorithm for. For this purpose we had access to a large database of anonymized real personal surnames–raw surnames were distilled and strictly separated from other personal data (forenames, birthdates, addresses, etc.). This database was originally established for testing purposes of other company IT projects. The database contains Slovak surnames and foreign surnames, originating mostly in middle and east European countries. Despite the anonymization we still cannot publish the data we were working with because of their origin.

From the database we extracted the surnames and created an initial data set for the algorithm development. This initial data set consists of 82,100 surnames. The portion of foreigners in the set is around 21%. In the context of our research project, foreign surnames are not typical, not expected and not often occurred in Slovakia, but they are usual in neighboring languages and countries (Czech, Polish, Ukrainian, Russian, German, Hungarian, Jewish).

The reason why we need a large set of surnames is to be able to choose good candidate queries for training and testing. Good surname - candidate for query can be determined by at least a few tens of differently spelled versions of the similar pronounced surname according to languages of our interest. Inspired by [7] we collected a set of hits retrieved by all of the implemented algorithms (including phonetics algorithms, N-Grams, Damerau-Levenshtein, etc.). As a new algorithm was implemented, the retrieved hits of the algorithm were added to the "to be judged" set–a set of hits, which needed to be evaluated for relevance. We set up a high distance threshold for Damerau-Levenshtein and small n-grams intersections count for n-grams of two compared words, which leads to a large number of retrieved hits by these algorithms.

The next step was an application of the Relevance Judgment technique. We did carefully assess each of the retrieved surnames and decided if it was either a relevant or not relevant hit according to phonetic of our target languages. We tried to assess the most common surnames of each target language. As mentioned before, we were searching for suitable query candidates–a surname with a sufficient amount of relevant hits (mostly at least 5 relevant) we added the query to the set of feasible surnames queries for the training set. In the second step we were trying to select from this group of queries the ones with surnames and associated hits in order to target a different phonetic error correction.

Later the MidEPhone rules were created using the following approach: First we have chosen rules from existing phonetic algorithms, the rules suitable for the phonetics of our languages. Second we created a new set of rules by considering a specific phonetics of languages targeted in our research, especially the Slovak language. Then we started adjusting and fine tune the rules to achieve the best result on the training set of the queries.

For the testing phase we injected another 2800 surnames extracted from genealogical databases, lists or registers of common surnames in this region, which were freely available on the internet to achieve a higher ration of foreign surnames in database. Hence the amount of this Test set of surnames increased to approximately 84,900 surnames and the proportion of foreigners increased to 23%. We then created a Test set of surname queries in a similar way as the training set for algorithm development. This time we used more people to judge the hit relevance to achieve a more objective view.

To compare MidEPhone with other phonetic algorithms, we have stored a canonical form of each surname for each phonetic algorithm, described in Section II-C. Similarly to [14] who tested the Phonix algorithm for different long canonical forms length; we stored into the database and then tested all the algorithms with the canonical forms of a maximum length of four, five, six and eight letters. The maximum length of canonical forms is labeled in figures with a number following an algorithm name. In the case of length 4 the number is omitted.

For summarization: The training set consists of 50 input queries and 308 asset relevant hits. The test set consists of 25

TABLE VI
SURNAMES TEST SET CLUSTERED BY THE ORIGINATED LANGUAGES

Surname	Associated relevant hits
Surnames of Polish origin	
Zahornaczky	Záhornacký, Zahoranský, Záhoranský
Koval'	Koval', Kovál', Koval, Kvál, Kovar, Kovař, Kobal, Kovář, Kovár, Chvál
Kruľák	Kruľák, Kruľak, Kuriľak, Kurylák, Kroliak, Kruliac, Kroliak, Krolák
Surnames of Slovak origin	
Kollár	Kollár, Kollar, Kullár, Kolár, Kolář, Klár, Gullár, Kullár, Gullár
Kováčik	Kováčik, Kovačik, Kovačik, Kvočik, Kavčík, Kováčech, Kuvíček, Kopačik, Kovacik, Krčmárek
Krčmárik	Krčmárik, Kačmárik, Krčmarik, Kečmárik, Krčmárek, Kramárik, Kačmarik
Krajčo	Krajčo, Krajča, Krajčov, Krajco
Ďuriač	Ďuriač, Ďurač, Ďurkač, Ďuriak, Buriak, Ďuriáš, Ďurjak, Ďuráč
Jurkovič	Jurkovič, Jurikovič, Jurčovič, Turkovič, Ďurkovič, Ďurovič, Jurčovič, Burkovič
Spišák	Spišák, Spiššák, Špišák, Spišak, Špišak, Spisak, Šišák, Pišák, Spišiak, Spišek
Surnames of Latin language origin	
Majtán	Majtán, Majtan, Majtáň, Majdán, Majan,
Škultéty	Škultéty, Škultéti, Skultety
Surnames of Jewish origin	
Solomon	Soloman, Salamon, Šalamon, Sólyom, Šalamon
Simeon	Simon, Semjon, Siman, Simeonov, Šimon, Šimun, Šimún, Šimún, Šimun
Surnames of German origin	
Bauer	Bauer, Bajer, Bayer, Sauer, Auer, Bayerl, Breuer, Daher, Mayer, Pauer, Paur, Paúr, Baur
Fišer	Fišer, Fizer, Ficzere, Fischer
Schmidt	Schmidt, Schmid, Šchmidt, Schmiedt, Schmied, Schmitz, Smidt, Šmidt, Smith, Šmíd
Schuster	Schuster, Schússer, Suszter, Fűrster, Szusztor, Šustr, Schússer, Šuster
Surnames of Czech origin	
Novák	Novák, Novak, Novek, Novak, Nowak
Svoboda	Svoboda, Sloboda, Szloboda
Dvořák	Dvořák, Dvorák, Dvořak, Dorák, Dvorčák, Dvornák, Dvorjak, Borák
Surnames of Hungarian origin	
Bodnár	Bodnár, Bodnar, Bognár, Bednar, Bednář, Bennár, Gondár, Bytnár, Gunár
Halás	Halás, Halas, Halász, Haás, Balász, Balázs, Galas, Haas, Halaš, Kalász Haász, Valas, Balazs, Balaž, Chalás, Chalas, Kalas, Hálász
Surnames of Ukraine origin	
Hricko	Hricko, Hrico, Hričko, Hrečko, Hirčko, Kričko, Hriczo, Hriško
Varchol	Varchol, Varchola, Varchoľ, Varhol, Vacho, Varhoľ, Barkol, Warchal

input queries and a total of 189 relevant hits associated to them, which are shown in Table VI.

A. Retrieved Hits Scoring

As the amount of records in the database grows the necessity to properly show a result becomes more significant. By having a large amount of data in the database, algorithms can possibly return a huge amount of records, as a result for some queries. [9] deals with the assessment of the individual hits retrieved by search engines. The collection of the documents returned as a result of the algorithm query is called a hitlist. A hitlist contains "good" and "bad" documents. The quality of the search engine is evaluated by the proportion of good documents, their position and the amount of missing "good" documents. The effort made to increase a proportion of "good" hits, placing them at the top of the hitlist and to reduce the amount of the missed "good" hits. Despite the principle of phonetic algorithms retrieval differs from the search engines, because they take in account the word phonetics; the ordering and scoring of the retrieved hits by the relevance is just as important when showing them to an end user.

Hits Distance Threshold: To assess "how good" the hit is we used the metric Damerau–Levenshtein distance. This metric is defined as a number of the operations: deletion, addition, replacement and transposition of the letter, needed to transform one word to another. Using this metric we were able to calculate the distance of the retrieved hits from the queried surname.

With a calculated distance we were able to perform the hitlist ordering or use it for algorithms scaling (to filter the retrieved hits by the maximum distance). For our testing of the average precision and precision recall graphs we used the hitlist ordered by the Damerau–Levenshtein distance.

B. Precision, Recall and Average precision

Precision and Recall are the most common metrics. Precision is the level of usability and Recall is the level of completeness of the records in a hitlist. Precision is defined by expression (3), recall by expression (4).

$$precision = \frac{relevant\ hits\ in\ hitlist}{hits\ in\ hitlist} \tag{3}$$

$$recall = \frac{relevant\ hits\ in\ hitlist}{relevants\ documents\ in\ collection} \tag{4}$$

A measurement which combines both precision and recall is called average precision. The average precision metric is the sum of the precision at each relevant hit in the hitlist divided by the total number of relevant documents in the hitlist [9]. Precision after m relevant hits is counted by function (5).

$$p_m = \frac{1}{m}\sum \frac{m}{k=1} x_k \tag{5}$$

Average precision is then calculated as expression (6), where R is a number of total relevant hits and n is the number of relevant hits in the hitlist.

$$p_m = \frac{1}{m}\sum \frac{m}{k=1} x_k \tag{6}$$

We used the metrics Precision, Recall, Average precision and Precision-recall graphs to test MidEPhone and other phonetic algorithms, described in Section II-C. We also tested different lengths of canonical forms including 4, 5, 6 and 8 letters length. We also tested non phonetic algorithms including variations of n-grams algorithm or other search techniques e.g. Hamming Distance, Jaro-Winkler metric, Exact match, Partial match. Due to the large number of combinations of tested algorithms we have chosen to present the result of the most adequate candidates, considering good results in all the mentioned metrics.

In the figures below we showed the result of the chosen algorithms tested on the test set of surnames. We decided to show the

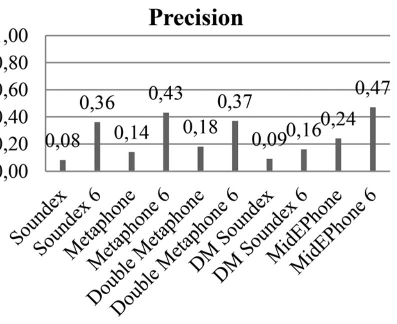

Fig. 2. The phonetic algorithms precision comparison.

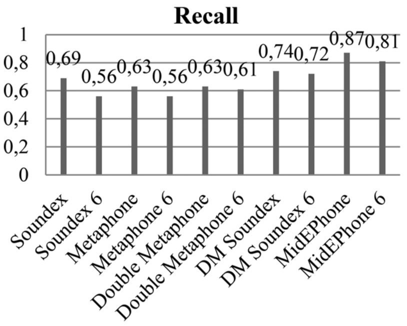

Fig. 3. The phonetic algorithms recall comparison.

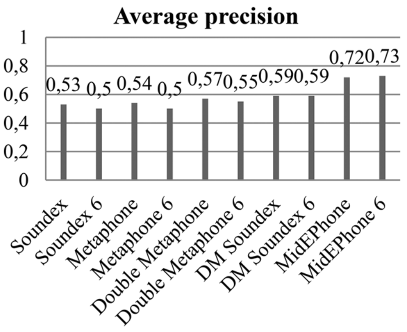

Fig. 4. Phonetic algorithm average precision comparison.

result for a canonical form of length 4 and 6 letters. In case of length 8, algorithms achieved significantly worse average precision as well as recall, but only slightly better precision. Therefore we do not consider this canonical form length results as of much significant for our purpose.

The results of the canonical form of 5 letters showed a tradeoff of precision and recall between the canonical form length 4 and 6. The lengths 6 and 4 are therefore good candidates to show the effect of the canonical form length scaling to the precision and recall of the hitlist. Suffix 6 after the algorithm name in the figures below, represents a variation of the algorithm with the canonical form of 6 letters length; no suffix means the 4 letters variation. The abbreviation DM Soundex is used for the Daitch-Mokotoff Soundex. Fig. 2 shows the comparison of the algorithms with measured precision.

Fig. 3 shows the good performance of MidEPhone and MidEPhone6 in comparison with the other algorithms using the recall measurement.

Fig. 4 shows the good performance of both MidEPhone algorithm variations also with the average precision measure.

TABLE VII
COMPUTED P-VALUE

	Soundex	**Soundex 6**	**Metaphone**	**Metaphone 6**	**Double Metaphone**	**Double Metaphone 6**	**DM Soundex**	**DM Soundex 6**
Precision	0,00	0,12	0,08	0,16	0,08	0,12	0,00	0,08
Recall	0,12	0,08	0,08	0,08	0,16	0,08	0,16	0,16
Average precision	0,08	0,04	0,08	0,04	0,12	0,04	0,12	0,12

C. Statistical Significance of the Precision, Recall and Average Precision Measurements

As shown before, the MidEPhone algorithm yields improvements using the metrics above in a retrieval evaluation, but now we want to know the statistically significance of the results [19]. Based on the results of our measurement we made a hypothesis, claiming the MidEPhone 6 to have better precision, recall and average precision compared to other phonetic algorithms (excluding the variation of the MidEPhone algorithm with a canonical form of length 4). We weighed the strength of this hypothesis by measuring the statistical significance using the p-value technique. For each algorithm (Soundex, Soundex 6, Metaphone …) we assumed a null hypothesis to reject: "The algorithm having a higher precision (resp. recall or average precision) than MidEPhone 6."

Table VII shows the p-value we computed on the Test set of the surnames.

Typically a statistical significance threshold 0,05 is used for rejecting the null hypothesis. In our case the measured values are close to the threshold level, moreover in some cases they fulfill the threshold. According to the result we cannot claim the result is in all cases statistically significant, however the practical importance of the result is still highly noteworthy. In some queries other algorithms achieved the same or slightly higher precision, recall or average precision than MidEPhone 6, but still in most cases MidEPhone algorithm yields the better result. Exactly in case of the *Precision* the results were better from 84% to 100% of the queries comparing to each of the algorithms in Table VII. In case of the *Recall*, the MidEPhone6 achieved better result from 84% to 92% of the queries and from 88% to 96% percent of the queries in case of the *Average precision* comparison. The fact that most of the queries achieved better measured results, while the others had same or slightly worse, is satisfactory enough for the essential goal of our research. For a practical usage in retrieval systems it is a substantial improvement.

D. Precision-Recall Graph

A frequently used technique to display the quality of a search is the precision-recall graph. The graph in Fig. 5 shows the degradation of precision against percentage recall. The graph shows a trade-off between precision and recall. Trying to increase recall typically introduces more bad hits into the hitlist, thereby reducing precision. Trying to increase precision typically reduces recall by removing some good hits from the hitlist.

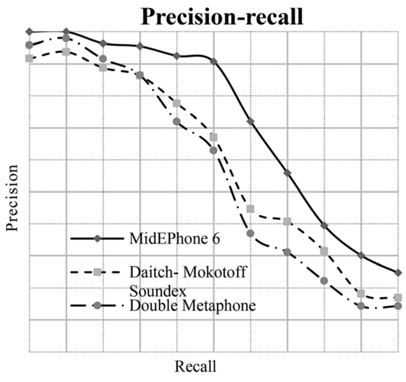

Fig. 5. Comparison of phonetic algorithms using the precision-recall graph.

The goal is to move the entire curve up and to the right which means that both recall and precision are better [9].

VI. Conclusions

During the research we did a study of existing phonetic algorithms. We could not find any publicly available algorithms with rules designed primarily for Slovak language. It is therefore possible that the proposed algorithm will be the first phonetic algorithm designed for the Slovak language.

The evaluation and comparison of phonetic algorithms is a difficult task, there is no method to prove that one algorithm is better than other. There is no strong evidence. However statistically based on our measurements, we can claim, that in the case of the phonetic matching of Slavic and surrounding countries surnames, MidEPhone has brought a significant increase in the search precision and better level of recall in comparison with algorithms, which claim a certain support for these languages too (Daitch-Mokotoff Soundex and Double Metaphone). Measurements of MidEPhone showed, that the algorithm can be a very feasible solution for search retrievals systems, which want to provide an advanced (phonetic errors aware) search capability of surnames in Slavic and some other Middle and East European languages.

MidEPhone implements an innovative principle of ending syllable transformation, which is an important part of Slavic surnames. The rules set of the algorithm were rigorously designed, based on the common surnames occurring in the region of middle Europe. The algorithm was trained and also tested on a large surnames dataset founded on real persons from the region. The algorithm offers the possibility to scale canonical form length. It also provides an option of querying a data store by matching preprocessed canonical forms, upon which indexes can be created to achieve a very effective search response.

Middle and east European languages contain diacritical characters. However all tested phonetics algorithms, including the MidEPhone, use rules set just for basic alphabet letters. Input words therefore need to be transliterated. As a result the diacritical letters are transformed into their basic equivalents. The diacritic itself changes the phonetic of the letter, thus transliteration loosens the phonetic information, because the word will be pronounced slightly differently as compared to the original. In our future work, we think it will be worthy to supply MidEPhone with additional rules, which will also cover diacritical letters transformation. We assume it will result in an even better precision, without decreasing the recall of the hitlist.

Acknowledgment

Special thanks go to Prof. Pavol Navrat for his suggestion and advice and to Gratex International for their support during the time of writing this paper.

References

[1] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios, “Duplicate Record Detection: A Survey,” *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 1, pp. 1–16, Jan. 2007.
[2] I. R. Draganov, A. A. Popova, and L. L. Ivanov, “Multilingual Names Database Searching Enhancement,” in *Proc. IEEE Int. Symp. Signal Process. Inf. Technol.*, Dec. 16–19, 2008, vol. 2008, pp. 474–479.
[3] T. Gadd, “Fisching fore werds: Phonetic retrieval of written text in information systems,” *Program: Automated Library and information systems*, vol. 22, no. 3, pp. 222–237, 1988.
[4] T. Gadd, “PHONIX: The algorithm,” *Program: Autom. Library and Inf. Syst.*, vol. 24, no. 4, pp. 363–366, 1990.
[5] G. Giannakopoulos, V. Karkaletsis, G. Vouros, and P. Stamatopoulos, “Summarization system evaluation revisited: N-gram graphs,” *ACM Trans. Speech Lang. Process.*, vol. 5, 3, p. 39, Oct. 2008, Article 5.
[6] D. Holmes and M. C. Mccabe, “Improving Precision and Recall for Soundex Retrieval,” in *Proc. Int. Conf. Inf. Technol.: Coding Comput.*, Washington, DC, Apr. 2002, pp. 22–26.
[7] M. Hosseini *et al.*, “Selecting a Subset of Queries for Acquisition of Further Relevance Judgements,” pp. 113–124, 2011, ICTIR, LNCS 6931.
[8] A. Kossoroukov, “Text Search and Fuzzy Matching,” Dec. 2005, Retrieved April 28, 2009, Search and Fuzzy Matching.ppt [Online]. Available: http://www.andreiko.com/designpatterns/Text
[9] K. Mahesh, *Text Retrieval Quality: A Primer. Oracle Technology Network*, 2001 [Online]. Available: http://oracle.com/technology/products/text/htdocs/imt_quality.htm, Oracle Corporation. (April 2009). Retrieved April 26, 2009
[10] G. Mokotoff, 2007, Soundexing and Genealogy. [Online]. Available: http://www.avotaynu.com/soundex.html.
[11] P. Odaloš, “Surnames in the context of history and present (social, territorial and ethnic determination),” *Slovak. Studia Slovaca*, pp. 205–212, 2004.
[12] M. K. Odell and R. C. Russell, “,” U.S. patents 1,261,167 (1918) and 1,435,683 (1922).
[13] B. T. Oshika, F. Machi, B. Evans, and J. Tom, “Computational techniques for improved name search,” in *Proc. 2nd Conf. Appl. Nat. Lang. Process., Appl. Nat. Lang. Conf. . Assoc. Comput. Linguist.*, 1988, pp. 203–210.
[14] U. Pfeifer, T. Poersch, and N. Fuhr, “Searching Proper Names in Databases,” In Hypertext - Information Retrieval - Multimedia, Synergieeffekte elektronischer Informations systeme, Proceedings HIM ’95. Univ. Konstanz, pp. 259–276, 1995.
[15] L. Philips, “Hanging on the Metaphone,” *Comput. Lang. 7*, vol. 12, pp. 39–43, 1990.
[16] L. Philips, “The Double Metaphone Search Algorithm,” C/C++ Users Journal 18 vol. 6, pp. 38–43, 2000.
[17] T. Kuzár, “Clustering on Social Web,” *Inf. Sci. Technol. Bull. ACM Slovakia*, vol. 5, no. 1, pp. 34–42, 2013.
[18] M. Samiec, “Surnames Phonetic Searching (In Slovak),” M.S. thesis, Faculty of Informatics and Information Technologies STU, Bratislava, Slovakia, 2007.
[19] M. D. Smucker, J. Allan, and B. Carterette, “A Comparison of Statistical Significance Tests for Information Retrieval Evaluation,” in *AProc. CM CIKM*, 2007, pp. 623–632.
[20] J. Soo and O. Frieder, “On Foreign Name Search,” ECIR, LNCS 5993 pp. 483–494, 2010.
[21] R. L. Taft, “Name search techniques, New York State Identification and Intelligence System,” Albany, NY, Special Rep. No. 1, 1970.
[22] C. Varol and C. Bayrak, “Hybrid Matching Algorithm for Personal Names,” *ACM J. Data Inf. Qual. 3*, vol. 4, p. 18, Sep. 2012, Article 8.
[23] E. M. Voorhees, “Common evaluation measures,” in *Proc. 12th Text Retrieval Conf. (TREC ‘03)*, 2003, pp. 1–13.

[24] P. Wallis and J. A. Thom, *Relevance Judgments for Assessing Recall, Inf. Process. Manage.*, vol. 32, no. 3, pp. 273–286, May 1996, ISSN 0306-4573.

[25] D. Zahoranský and I. Polasek, "Rule Based Phonetic Search Approach for Central Europe, SISY 2010," in *Proc. IEEE Int. Symp. Intell. Syst. Inf.*, Subotica, Serbia, 2010, pp. 71–76.

[26] J. Zobel and P. Dart, "Phonetic string matching: Lessons from information retrieval," in *Proc. 19th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, Zurich, Switzerland, 1996, pp. 166–172.

[27] J. Zobel and P. Dart, Fnetik: An integrated system for phonetic matching Dept. of Comput. Sci., RMIT, Tech. Rep. 96-6, 1996.

Dušan Zahoranský received the Engineer degree from the Faculty of Informatics and Information Technology, Slovak University of Technology (FIIT STU) in Bratislava in 2010, from the field of software engineering. His area of research and master thesis were focused on phonetic search algorithms. Since 2007, he has been a Programmer in the Software Development Division of Gratex International. His work includes development of enterprise systems for information retrieval and registries.

Ivan Polasek received the Ph.D. degree in applied informatics in 2000 and since 2003 he has been an Assistant Professor at the Faculty of Informatics and Information Technology, Slovak University of Technology (FIIT STU) in Bratislava lecturing on software architectures and object oriented analysis and design. In 1990, he worked or cooperated with NCR Germany, SCD Wien, Informata Zurich, Gesellschaft für Angewandte Informatik Bern and since 1997 he has been a Project Manager and a Head of the Software Analysis and Design Group in the software company Gratex International.