Approximate String Matching for Geographic Names and Personal Names

Clodoveu A. Davis Jr.[1], Emerson de Salles[1]

[1]Instituto de Informática – Pontifícia Universidade Católica de Minas Gerais
Rua Walter Ianni, 255 – 31980-110 – Belo Horizonte – MG – Brazil

clodoveu@pucminas.br, hemer80@gmail.com

***Abstract.** The problem of matching strings allowing errors has recently gained importance, considering the increasing volume of online textual data. In geotechnologies, approximate string matching algorithms find many applications, such as gazetteers, address matching, and geographic information retrieval. This paper presents a novel method for approximate string matching, developed for the recognition of geographic and personal names. The method deals with abbreviations, name inversions, stopwords, and omission of parts. Three similarity measures and a method to match individual words considering accent marks and other multilingual aspects were developed. Test results show high precision-recall rates and good overall matching efficiency.*

1. Introduction

The problem of matching strings allowing errors and other kinds of discrepancies has been studied for some time (Navarro 2001), but still presents some interesting and challenging issues. Its importance is growing, since large volumes of textual data have become available through the Internet. Applications of approximate string matching nowadays include some important areas in computing, such as information retrieval, digital libraries, ontology integration and computational biology. In many of those applications, such tools are needed whenever the input data are uncertain, semi-structured, or simply reflect the various views people have on their surrounding environment.

Approximate string matching techniques are frequently required in geotechnologies and in applications that have to deal with much semantic uncertainty. Consider, for instance, the response of a person to the simple question "Where do you live?" Depending on the context of the conversation, the answer may be the name of a city, a state, a country, or even an address, with details such as a house number and a postal code. Anyway, the description certainly includes references to one or more place names, which can be ambiguous and contain subtle and misleading errors. Many ambiguities result from the reuse of the names of places that are famous elsewhere ("Paris, Texas", as opposed to "Paris, France"), while errors may be caused by spelling difficulties (place names that include foreign words or proper nouns), language or local particularities (phonetic alphabets and use of accent marks). String matching is also needed in geographic ontology integration, geographic information retrieval (Jones, Purves et al. 2002; Borges, Laender et al. 2007), and other semantic-related subjects.

Our interest in approximate string matching lies on the applications that use place names and their variations. Since personal names are frequently used as place names, our interest extends towards them as well[1]. Research on retrieval of personal names and proper nouns is quite active, with many recent works (Barcala, Vilares et al. 2002; Cohen, Ravikumar et al. 2003; Patman and Thompson 2003; Minkov, Wang et al. 2005; Christen 2006). Previous work on gazetteers (Souza, Davis Jr. et al. 2005), geographic information retrieval (Borges, Laender et al. 2003; Delboni, Borges et al. 2007), geo-coding (Davis Jr., Fonseca et al. 2003) and address matching (Davis Jr. and Fonseca 2007) has shown that the use of approximate matching algorithms can be an important asset whenever the input data have been manually fed or obtained directly from natural language text. This work, therefore, presents string matching techniques that seek better results and more flexibility when dealing with geographic and personal names.

The remainder of this paper is organized as follows. Section 2 introduces the approximate string matching problem in more detail and discusses existing techniques. Section 3 studies specific aspects of matching personal and geographic names, and presents our approach to the problem. Section 4 shows experimental results. The paper is concluded in Section 5, which also presents some of our priorities for ongoing work.

2. Problem statement and related work

The problem of approximate string matching is a traditional one in computer science. For recent surveys of initiatives on this subject, see (Navarro 2001) and (Christen 2006). In an informal way, approximate string matching corresponds to finding out if a given string *S* matches a pattern *P*, within a given similarity threshold δ. This threshold can be given as a maximum number of allowable errors, or as a normalized measure, a number between 0 and 1, with 0 meaning a mismatch and 1 meaning an exact match.

Since we are interested in personal and geographic names, our review of related work pays more attention to techniques that work better when both the string and the pattern are short. Some approximate string matching techniques and algorithms are more suited to seek the presence of a short pattern in a (presumably long) text. Given the objective of this paper, we will not discuss this variation further. However, we only point out that this kind of matching is important for geographic information retrieval, in algorithms that try to determine the geographic context in a natural-language text, based on landmark names and on expressions that indicate locality (Borges, Laender et al. 2007).

More formally, approximate string matching can be stated as in Definition 1.

> ***Definition 1.*** *Let S and P be strings of characters, i.e. sequences of symbols from a finite alphabet* Σ. *P is called the pattern, against which S will be compared. We say that S matches P when* $f(S,P) < \delta$, *where* $f : \Sigma \times \Sigma \rightarrow [0,1]$ *is a distance function, which quantifies the similarity between S and P and* δ *is a similarity threshold.*

In other words, *f* indicates how close *S* is from *P*. Several metrics have been proposed for the similarity, depending on the algorithm (Cohen, Ravikumar et al. 2003).

There are two main types of approximate matching techniques: *phonetic matching* and *pattern matching*. Phonetic matching considers the similarity of letters in a string as to the usual sounds they produce in English, and generates a code from any given word. In

[1] An assessment of the street names catalog for the city of Belo Horizonte, Brazil, showed that about 67% of the names have more than one word. Of those, about 65% are references to personal names.

general, similarly sounding words produce equal codes, but subtle spelling differences can keep phonetic matching from identifying many similarities. With these and other limitations, some studies (Zobel and Dart 1995; Christen 2006) have shown that phonetic matching is systematically outperformed by pattern matching techniques as to its matching efficiency, even though phonetic methods tend to run faster. There are also hybrid methods, combining string distance measures and probabilistic interpretations of pattern matching results (Pfeifer, Poersch et al. 1996; Cohen, Ravikumar et al. 2003).

Pattern matching relies on comparing characters from each string, to detect points in common and to quantify their similarity based on that. Some of the most important pattern matching techniques are based on edit distance. One of such techniques (Levenshtein 1965) determines the total number of operations (insertions, deletions, or substitutions) required to transform *S* into *P*. This number is called the *edit distance* or the *Levenshtein distance* (*LD*) between *S* and *P*. Considering that each operation has a cost of 1, similarity can be calculated as

$$f_{LD}(S,P) = 1.0 - \frac{LD(S,P)}{\max(|S|,|P|)} \tag{1}$$

Variations of LD consider the transposition of characters as another operation, with unit cost, or costs that depend on similarity criteria applied to individual characters (Navarro 2001). For instance, for optical character recognition, the comparison between similar letters such as “i” and “l” can have a lower cost. For manual input, a lower substitution cost can be assigned to pairs of letters that are adjacent on the keyboard.

The Levenshtein edit distance algorithm is usually implemented using a matrix $L[|S|+1,|P|+1]$ of integers, filled out in a row-wise traversal to the right, as in Eq. 2:

$$L_{i,j} = \begin{cases} L_{i,0} = i \\ L_{0,j} = j \\ if\_S_i = P_j\_then\_L_{i,j} = L_{i-1,j-1} \\ if\_S_i \neq P_j\_then\_L_{i,j} = \min(L_{i-1,j}, L_{i,j-1}, L_{i-1,j-1}) + 1 \end{cases} \tag{2}$$

As a result, $LD(S,P) = L_{|S|,|P|}$. If the “+1” clause at the last row of equation 2 is changed to a function, it is possible to consider varying costs according to the type of operation or to the similarity between pairs of characters.

Another similarity metric, which is not based on an edit distance model, has been proposed by Jaro and later refined by Winkler (Jaro 1995; Winkler 1999). The Jaro-Winkler algorithm is bases on the number of common characters and the order in which they appear, increasing the similarity value in case there is a match on the initial characters of the string. We decided not to use this metric, partly because some works have shown that their results are better suited to matching short strings (for instance, last names) (Cohen, Ravikumar et al. 2003) and because the Levenshtein distance would suit our ideas for multiword strings better. There is, however, an adaptation of the Winkler technique in which all possible permutations of the words in a multiword string are considered (Christen 2006). Our method can deal with inversions, but is able to avoid performing the permutations, as we will show next.

3. Matching Personal and Geographic Names

Matching personal or geographic names can be defined as the process of determining, within a given level of certainty, whether two strings correspond to the same person or place. Matching names is a challenging task, mainly because of spelling variations and some widely used practices, such as abbreviation. This is further complicated by the adoption of different ordering and by the varying importance assigned to parts of the name, based on cultural differences (Patman and Thompson 2003). Such variations usually keep exact matches from being effective. Spelling variations are common in personal names, both in given names and surnames, as well as in cross-language adaptations of place names. For instance, London is referred to as "Londres" in Portuguese, while "Lisboa" is referred to as "Lisbon" in English. Spelling variations also arise from the transliteration of words and names from different alphabets, as in Chinese, Russian, and other languages, to the Roman alphabet. Another source of variations is the intentional abbreviation and inversion of names, as in the indication of authorship in bibliographic references. In such sources, parts of names are also occasionally omitted.

Spelling variations often cause problems when, for instance, names are dictated over the phone for manual input. Christen (2006) includes both spelling variations and manual input among the primary sources of errors in personal names. Errors on common words can be detected using a dictionary, but detecting errors on names requires a different approach. In spite of spelling and presentation format difficulties, personal names are frequently used to designate places. Even though we do not have detailed data on that, we suspect, from personal observation in Brazil and abroad, that personal names are present on a large share of urban place names. Naming geographic features after people as a form of homage is an established tradition, hence names such as *Magellan strait*, *Weddell sea*, or *Hudson bay*, and, more recently, names assigned to features in other planets, such as *Tycho crater*, on the Moon, or *Maxwell Montes*, on Venus.

We observe that personal and geographic names share some common characteristics. Words are usually small, ranging from three or four characters up to no more than a hundred characters. Special characters, such as accent marks, are used depending on the language, and sometimes are omitted. Abbreviations are common, mostly on middle names. Words can occasionally be inverted or even omitted. Stopwords (such as "of" in English, or "de" in Portuguese) are often present in full names, but are frequently omitted in practice. Titles or professional descriptions are used along with personal names, usually preceding them (as in "Presidente Kubitschek"), and titles can also be abbreviated ("Pres. Vargas"). Such observations are mostly from practice, but we have been able to confirm most of them during our experiments on preliminary test data.

In the next section, we will present our approach to matching personal and geographic names, in which we address all of the above characteristics, by adapting existing techniques and adding heuristics of our own.

3.1 Matching single words

For single words matching, we implemented a variation of the Levenshtein edit distance method. Two word strings are considered to match if their similarity measure f is greater than a threshold δ (Eq. 1). We can determine whether this similarity can be reached using two pre-tests in sequence. If the difference in length of S and P exceeds the allowed number of errors, no further comparison is necessary, and *S* cannot match *P*. In other words, if $abs(|S|-|P|)/\max(|S|,|P|) > 1.0-\delta$ there is a mismatch on the basis of the *length test*. The other test is based on the *bag distance* (Bartolini, Ciaccia et al. 2002), a linear-time test that generates a lower bound for the edit distance. If we put together two sets of individual characters, *X* and *Y*, each of which formed with the character from *S* and *P*, respectively, then $bag(S,P)=\max(|X-Y|,|Y-X|)$, where the minus sign indicates the set difference operation. It has been shown that $bag(S,P)\le LD(S,P)$, and therefore we can discard the possibility of a match whenever $bag(S,P)/\max(|S,|P|) > 1.0-\delta|$ (*bag test*) (Bartolini, Ciaccia et al. 2002). The bag test allows us to discard mismatches before determining the actual edit distance at a much higher computational cost, as shown next.

Our variation of the Levenshtein edit distance calculation consists in incorporating a practical scheme for matching accent-marked letters and special characters. Characters are organized into groups, so that characters belonging to the same group are considered to match, as formally described in Definition 2.

> ***Definition 2.*** *Let c_1 and c_2 be two characters from the alphabet Σ, and let $g_1, g_2, \ldots, g_n$ be n sets of at least 1 character, in which $g_i \cap g_j = \varnothing, \forall i \neq j$. We say that c_1* matches c_2 under g_i *when there is a group g_i which contains both c_1 and c_2, i.e.,* $c_1 \overset{g_i}{\equiv} c_2 \Leftrightarrow g_i \supset \{c_1, c_2\}$

Thus, equivalent characters are organized into groups, as shown in Table 1. This strategy allows us to prepare several sets of groups, according to the requirements of a matching effort and the characteristics of source and pattern data. For instance, case-sensitive groups, phonetic groups, or accent-mark-sensitive groups can be prepared and used with no code modification. Figure 1 shows examples of case- and accent-mark-sensitive (a) and insensitive (b) matching using the Levenshtein matrix, in the comparison of the names of two Brazilian cities, Paranaguá (PR) and Paranapuã (SP).

Table 1 - Groups of characters

Id	*Group*	*Id*	*Group*
L1	a, á, ã, à, ä, â	U1	A, Á, Ã, À, Ä, Â
L2	e, é, è, ë, ê	U2	E, É, È, Ë, Ê
L3	i, í, ì, ï, î	U3	I, Í, Ì, Ï, Î
L4	o, ó, õ, ò, ö, ô	U4	O, Ó, Õ, Ò, Ö, Ô
L5	u, ú, ù, ü, û	U5	U, Ú, Ù, Ü, Û
L6	n, ñ	U6	N, Ñ
L7	c, ç	U7	C, Ç
...	one group for each consonant (lowercase)	...	one group for each consonant (uppercase)

		P	**a**	**r**	**a**	**n**	**a**	**g**	**u**	**á**
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
P	**1**	0	1	2	3	4	5	6	7	8
a	**2**	1	0	1	1	2	2	3	4	4
r	**3**	2	1	0	1	2	3	3	4	5
a	**4**	3	1	1	0	1	2	3	4	4
n	**5**	4	2	2	1	0	1	2	3	4
a	**6**	5	2	3	1	1	0	1	2	2
p	**7**	6	3	3	2	2	1	1	2	3
u	**8**	7	4	4	3	3	2	2	1	2
ã	**9**	8	4	5	3	4	2	3	2	$\boxed{\mathbf{1}}$

(a) $f_{LD} = 0.89$

		P	**a**	**r**	**a**	**n**	**a**	**g**	**u**	**á**
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
P	**1**	0	1	2	3	4	5	6	7	8
a	**2**	1	0	1	1	2	2	3	4	5
r	**3**	2	1	0	1	2	3	3	4	5
a	**4**	3	1	1	0	1	2	3	4	5
n	**5**	4	2	2	1	0	1	2	3	4
a	**6**	5	2	3	1	1	0	1	2	3
p	**7**	6	3	3	2	2	1	1	2	3
u	**8**	7	4	4	3	3	2	2	1	2
ã	**9**	8	4	5	3	4	2	3	2	$\boxed{\mathbf{2}}$

(b) $f_{LD} = 0.78$

Figure 1 - Accent-mark-insensitive (a) and sensitive (b) edit distance

In the definition of the groups and in the implementation, we used the Unicode standard set of characters. Notice that most information retrieval and text mining efforts usually pre-process the input strings, eliminating uppercase characters, accent marks and other special characters. We decided not to do so, since uppercase is often used as a way to distinguish proper nouns, and since accent marks can be decisive in determining a match, depending on the language.

One possible difficulty in using our method is the determination of the similarity threshold δ. We can understand more easily how to choose a value for δ if we think in terms of number of allowable errors. Considering S and P to have the same length, the maximum (integer) number of allowable errors is equivalent to $\lfloor (1.0-\delta)\cdot|S| \rfloor$. In order illustrate that, we performed a frequency distribution analysis of word lengths in a data set containing 9,222 personal names, extracted from BDBComp (Brazilian Digital Library on Computing)[2]. After separating 26,924 words from names, we observe that most names have between two and four words. Almost 75% of them have between 4 and 9 characters (Table 2). The large number of 1-character words reflects the use of abbreviations in BDBComp. Therefore, using a threshold $\delta = 0.75$, it means we allow for a maximum of one error in a 4- to 7-letter word, and 2 errors for a 8- to 11-letter word. This threshold seems adequate for personal names, and is left here as a suggestion. In our method for multiword matching, presented in the next section, this threshold applies to individual words taken separately, not to the full string.

Table 2 – Frequency distribution of BDBComp name lengths and number of words

Length	# names	%
1	4383	16,3
2	1377	5,1
3	469	1,7
4	2080	7,7
5	4688	17,4
6	4727	17,6
7	4638	17,2
8	2587	9,6
9	1281	4,8
10	426	1,6
11	182	0,7
12	52	0,2
13	19	0,1
14	8	0,0
15	7	0,0
TOTAL	26924	100,0

# words	#names	%
2	3894	42,2%
3	2922	31,7%
4	1760	19,1%
5	553	6,0%
6	81	0,9%
7	11	0,1%
8	1	0,0%
TOTAL	9222	100,0%

3.2 Matching multiword strings

The first step for matching multiword strings is dividing them into words, using whitespace characters as delimiters, such as blanks, hyphens and other symbols. Points are preserved as the last character in the preceding word, since they can indicate abbreviations. We can also opt to preserve or to eliminate stopwords. Stopwords constitute another way to differentiate between very similar names, and therefore we prefer to preserve them in most situations. However, some sources intentionally leave them out, as in the case of names in bibliographic references, so our implementation allows treating or discarding stopwords as an option. Our matching strategy then proceeds in four phases: (1) checking for standard abbreviations, (2) checking for non-standard abbreviations, (3) word-by-word matching and (4) verifying inversions.

[2] http://www.lbd.dcc.ufmg.br/bdbcomp/bdbcomp.jsp

First, each word in the string is tested against a list of known abbreviations. The list contains pairs of the type <*abb*, *val*>, where *abb* is the standard abbreviation (for instance, "Pres"), and *val* is its meaning, spelled completely (as in "President"). If an *exact* match is found between a word in *S* and an abbreviation from the list, it is replaced by its full spelling. Our intention is to expand abbreviated titles, which are quite common preceding personal names and in some kinds of place names, to their full description. We do not expect to find many false matches, i.e., words that coincide with standard abbreviations but have a meaning of their own (as in someone whose name is "Pres"). The possibility of such coincidences should be assessed by the user, who could then leave conflicting abbreviations out of the list.

Next, non-standard abbreviations are verified. Candidates are 1-character capitalized words and words that end with a point. Such abbreviations are compared to each word in the pattern, and a similarity measure is then calculated as the number of matching characters of the abbreviation $S[i]$ divided by the number of characters in the candidate word from the pattern $P[j]$, where $X[k]$ denotes the kth word of string *X* (Eq. 3).

$$f_{NSA}(S[i], P[j]) = \frac{|S[i]|}{|P[j]|} \tag{3}$$

Unless the similarity threshold is set too low, any non-standard abbreviations in *S* will not find a match with regular names. We assume, heuristically, that the abbreviation has been used in order to save space or typing effort; therefore, we expect a large difference in size between the abbreviated word and its expected match in the pattern. On the other hand, it is likely that a non-standard abbreviation will reproduce the first characters from the corresponding word. We consider this case to be a match if all the characters in the abbreviated word are *equal* (i.e., no approximation is allowed) to the same number of characters at the beginning of the pattern word, which is where the characters that form an abbreviation are usually taken from. Even though in most cases name abbreviations involve simply an initial, with this heuristic we expect to be able to match unusual abbreviations or abbreviations that have been left out of the standard list.

In the third stage, we perform word-by-word matching, using a strategy that is similar to LD calculation (Eq. 2), modified to allow for inversions and to provide a similarity measure. Our matching algorithm uses a matrix $W\left[|S|_W, |P|_W\right]$, where $|X|_W$ denotes the number of words in string X. The matrix is filled out in a row-wise traversal to the right, making $W_{i,j} = f_{LD}(S[i], P[j])$ if $S[i]$ is a regular name, or $W_{i,j} = f_{NSA}(S[i], P[j])$, if $S[i]$ is a non-standard abbreviation. The value of f_{LD} is determined using the process described in the previous section. When $S[i]$ is a name, after each row *i* is complete, we identify the column *j* at which the value of the similarity function is maximum. If this value exceeds the similarity threshold δ, a match exists between $S[i]$ and $P[j]$. For the processing of the next row, the word $P[j]$ is left out of the similarity comparisons if the match was exact. At the end of the process, we select the best match for each word in the pattern, considering a valid match only when (1) the similarity threshold has been reached, or (2) there is a match with a non-standard abbreviation.

Denoting as *v* the number of matching words, we propose three similarity measures for multiword string matching. The first, f_{MW}, is calculated dividing the sum of the similarity values found for each matching word by the number of matching words, giving an idea about the average similarity of words in each string (Eq. 4). The second, f_{VM}, indicates the fraction of the words from the pattern for which a match has been found (Eq. 5). The third measure, f_{INV}, indicates the occurrence of inversions, and is calculated as follows. The order in which words from the string match words from the pattern is generated and analyzed, counting the number of times in which the sequence is broken. The f_{INV} similarity is then calculated by establishing a penalty for each inversion, corresponding to the number of inversions (n_I) divided by the number of matching words (Eq. 6).

$$f_{MW}(S,P) = \frac{\sum_{j=1}^{|P|_W} \left[\max_{i=1}^{|S|_W} \left(f_{LD}(S[i],P[j]),\ f_{NSA}(S[i],P[j]) \right) \right]}{v} \tag{4}$$

$$f_{VM}(S,P) = \frac{v}{\max\left(|S|_W, |P|_W\right)} \tag{5}$$

$$f_{INV}(S,P) = 1.0 - \frac{n_I}{v} \tag{6}$$

The similarity values can be used separately or combined with a weighted average. Weights are assigned according to the characteristics of the matching effort. For instance, when matching full names to bibliographic references, inversions are expected, so the inversion index can receive a lower weight than the other two measures. Equation 7 shows the calculation of the overall similarity f, where w_{MW}, w_{VM}, and w_{INV} are respectively the weights for word, valid matches, and inversions, and $w_{MW} + w_{VM} + w_{INV} = 1$.

$$f(S,P) = w_{MW} f_{MW}(S,P) + w_{VM} f_{VM}(S,P) + w_{INV} f_{INV}(S,P) \tag{7}$$

Figure 2 shows the comparison of two names, considering $\delta = 0.75$, case- and accent-mark-sensitivity. In the first row, the only the first words match, with a similarity of 0.857 (one error in seven characters). The other words from the pattern do not match the first word from the string; a similarity measure does not have to be calculated, since the comparisons fail either the length test or the bag test. Further comparisons only have to be made on "Antônio", first word from the pattern, if an exact match has not been found. In the second row, "C." is a non-standard abbreviation, and is compared against each word from the pattern. A match occurs with "Carlos", for which "C." is a possible initial. However, "Carlos" is not taken out of future comparisons, since a better match can occur with some other word. In the remaining two rows, the edit distance has to be calculated only twice. Since all words from the pattern found a match, $f_{VM} = 1.0$, and $f_{MW} = 0.706$, the average of the values in bold in Figure 2. Matches are in order (1<2<3<4), so $f_{INV} = 1.0$ also.

	Antônio	**Carlos**	**de**	**Souza**
Antonio	**0.857**	*	**	*
C.	**	**0.167**	0.000	0.000
de	**	**	**1.000**	**
Sousa	*	*	***	**0.800**

(*) discarded: bag test; (**) discarded: length test; (***) discarded: previous match

Figure 2 -- Multiword string matching example

Figure 3 shows a similar example. The value for f_{MW} is 0.664, indicating a penalty for the unmatched second name. The value for f_{VM} is now 0.75, for three matches out of four words. No inversions are found (1<3<4), so $f_{INV} = 1.0$.

	Antônio	**Carlos**	**de**	**Souza**
Antonio	**0.857**	*	**	*
Coelho	*	*	**	*
de	**	**	**1.000**	**
Sousa	*	*	***	**0.800**

(*) discarded: bag test; (**) discarded: length test; (***) discarded: previous match

Figure 3 – Another multiword string matching example

Figure 4 shows an example in which stopwords have been omitted from the string, but are present on the pattern. There are also inversions, as in a bibliographic citation. The value of f_{MW} is 0.278, a low value caused by the uncertainty associated with the matching of the initials. Since there are again three matches in four words, f_{VM} is 0.75. One inversion is found (4>1<3), so $f_{INV} = 0.67$.

	Antônio	**Carlos**	**de**	**Souza**
Sousa	**	*	**	**0.800**
A.	**0.143**	0.000	0.000	**
C.	0.000	**0.167**	0.000	0.000

(*) discarded: bag test; (**) discarded: length test; (***) discarded: previous match

Figure 4 -- Matching with inversions

Similarity values can be used to rank the strings against the pattern, either individually or by combining the measures, as in Equation 7. Table 3 shows the overall similarity values for the three examples, considering equal weights for w_{MW}, w_{VM} and w_{INV}.

Table 3 – Similarity values compared

	f_{MW}	f_{VM}	f_{INV}	f
Antonio C. de Sousa	0.706	1.000	1.000	**0.902**
Antonio Coelho de Sousa	0.664	0.750	1.000	**0.805**
Sousa, A. C.	0.278	0.750	0.670	**0.566**

4. Experimental results

We performed two experiments to assess the efficiency of the proposed method. One compared a bibliographic database to a list of personal names from a Web page, and the other compared manually input street names to an official thoroughfare catalog.

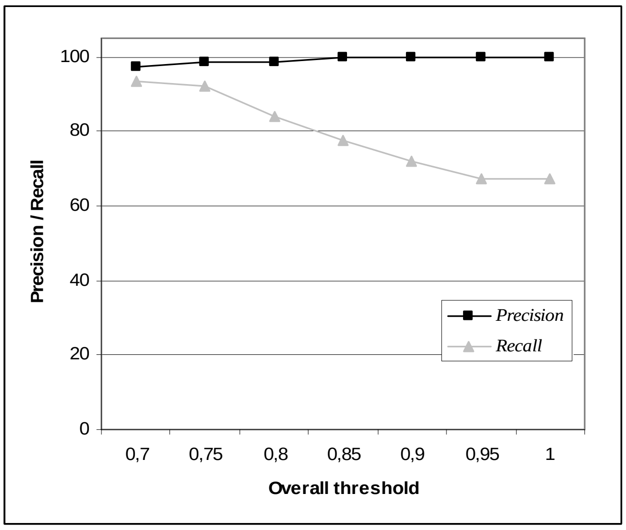
4.1 First experiment: personal names

A list of 85 personal names of Brazilian researchers on Computer Science has been extracted from a Web page in which the results of a grant bid (Edital 01/2007[3]) were published. Names from this list were compared against 9,222 author names from the already mentioned Brazilian Digital Library on Computing (BDBComp), which currently holds data on over 5,200 works published in national journals or conferences. Since both sources contain names of people from the same research community, we expected many matches, i.e., people that received a grant would probably have some paper published in a Brazilian journal or conference. Since CNPq names probably came from the Brazilian national curriculum vitae database, we expected full correctness, because names are ultimately input by the researchers themselves. Manual inspection showed

[3]http://efomento.cnpq.br/efomento/divulgacao/divulgacaoResultados.do?metodo=propostas&codigoLinhaFomento=58&seqChamada=17&idComite=CC

several minor and possibly intentional accent mark oversights, but no abbreviations. In BDBComp, however, abbreviations are common, since it is a bibliographic database.

CNPq names were matched against BDBComp names under the following setup: (1) inversions were not considered (f_{INV} = 1.0 in all comparisons), (2) stopwords, accent-mark- and case-sensitivity were enabled, and (3) $\delta = 0.85$. An overall similarity value *f(S,P)* was calculated as the simple mean of f_{MW}, f_{VM} and f_{INV}. Matches found were then manually checked for false positives and false negatives. Results were summarized using the precision and recall metrics from Information Retrieval (Baeza-Yates and Ribeiro-Neto 1999). Figure 5 shows a precision-recall graphic with *f(S, P)* varying from 0.70 to 1.00. Precision indicates the percentage of correct matches within all matches obtained, while recall indicates the percentage of expected matches that was achieved by the method. Notice that, as the threshold increases, precision rises (i.e., only closer matches are accepted), but recall drops.



f(S, P) threshold	Precision (%)	Recall (%)
0,70	97,3	93,4
0,75	98,6	92,1
0,80	98,5	84,2
0,85	100,0	77,6
0,90	100,0	72,0
0,95	100,0	67,1
1,00	100,0	67,1

Figure 5 - Precision/recall results

Considering *f(S, P)* as a similarity measure, we allowed as many matches names within BDBComp as possible, within the threshold. With this, in most cases multiple matches were obtained. In the lower threshold experiments, some names correctly matched as many as seven BDBComp names, indicating the occurrence of many spelling and abbreviation variations of the same person's name in BDBComp. Figure 6 shows the variation of the average number of correct and incorrect matches per CNPq name. These results indicate that our method could be used to cluster variations of the same name in BDBComp, thus improving the results of author queries to the digital library.

f(S, P) threshold	Correct	Incorrect
0,70	2,77	0,89
0,75	2,77	0,41
0,80	2,38	0,05
0,85	2,08	0,00
0,90	1,69	0,00
0,95	1,22	0,00
1,00	1,20	0,00

Figure 6 – Average number of correct and incorrect matches per name

4.2 Second experiment: geographic names

In a second experiment, we compared a set of a hundred street names against Belo Horizonte's official thoroughfare catalog. The street names were randomly selected from a list of 4,700 manually typed records, as part of a data collection effort. Street names from the list included many problems for geographic name matching, such as abbreviations, omission of parts, and misspellings. All selected street names and were manually geocoded by technicians from PRODABEL, the municipal IT company, who have much experience with local place names. They were allowed to use other data to resolve ambiguities manually. As a result, 87 of the 100 street names were recognized, while the remaining 13 either were unrecognizable, or from nearby cities.

Street names were then matched against the thoroughfare catalog using our method, with case- and accent-mark insensitiveness, stopwords and abbreviations considered, inversions allowed, and $\delta = 0.85$. We used the full street name string, including the thoroughfare type ("Rua", "R.", "Av.", and so on). Under these parameters, 62 of the 87 names were matched correctly, and only 4 were incorrect matches. We achieved a precision of 94% and a recall of 71%, which is similar to the results in Figure 5, but parsing out the thoroughfare type should lead to better results. However, if we considered only the street name, and allowed for case and accent marks, only 32 correct matches would remain, and the recall rate would drop to 35%. Allowing for approximate word matches, inversions, and abbreviations doubled the recall rate in this experiment.

5. Conclusions and future work

Approximate string matching for personal and geographic names is an important and useful technique, with applications in geographic information science, information retrieval, and other areas. We are particularly interested in using the proposed method for multilingual gazetteers, geocoding, geographic information retrieval, and records linkage. The adaptations we propose for individual word matching considering case- and accent-mark-sensitivity can also be used in applications such as multilingual ontology integration and natural language processing.

Our experiments, although preliminary, have demonstrated the validity of the proposals and ideas presented in this paper. We consider this line of work promising, even though more tests with a larger volume of data are required, in order to adequately assess the computational efficiency of the method and to compare it to other proposals. An experiment in records linkage, involving large volumes of data from the health sector, is being prepared. The integration of the techniques presented in this paper to a previous work (Davis Jr. and Fonseca 2007) is also planned.

References

Baeza-Yates, R. and Ribeiro-Neto, B. (1999). Modern Information Retrieval, Addison-Wesley.

Barcala, F. M., Vilares, J., Alonso, M. A., Graña, J. and Vilares, M. (2002). Tokenization and proper noun recognition for information retrieval. 13th International Workshop on Database and Expert Systems Applications, 246-250.

Bartolini, I., Ciaccia, P. and Patella, M. (2002). String Matching with Metric Trees Using an Approximate Distance. Proceedings of the 9th International Symposium on String Processing and Information Retrieval, Springer-Verlag.

Borges, K. A. V., Laender, A. H. F., Medeiros, C. B., Silva, A. S. and Davis Jr., C. A. (2003). The Web as a Data Source for Spatial Databases. V Brazilian Symposium on GeoInformatics (GeoInfo 2003), CD-ROM, Campos do Jordão (SP).

Borges, K. A. V., Laender, A. H. F., Medeiros, C. M. B. and Davis Jr., C. A. (2007). Discovering geographic locations in Web pages using urban addresses. 4th Workshop on Geographic Information Retrieval, to appear, Lisbon, Portugal.

Christen, P. (2006). A Comparison of Personal Name Matching: Techniques and Practical Issues. Sixth IEEE International Conference on Data Mining - Workshops, 290-294, Hong Kong, IEEE.

Cohen, W. W., Ravikumar, P. and Fienberg, S. E. (2003). A Comparison of String Distance Metrics for Name-Matching Tasks. Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web, 73-78.

Davis Jr., C. A. and Fonseca, F. T. (2007). "Assessing the Certainty of Locations Produced by an Address Geocoding System." Geoinformatica 11(1): 103-129.

Davis Jr., C. A., Fonseca, F. T. and Borges, K. A. V. (2003). A Flexible Addressing System for Approximate Urban Geocoding. V Brazilian Symposium on GeoInformatics (GeoInfo 2003), CD-ROM, Campos do Jordão (SP).

Delboni, T. M., Borges, K. A. V., Laender, A. H. F. and Davis Jr., C. A. (2007). "Semantic Expansion of Geographic Web Queries Based on Natural Language Positioning Expressions." Transactions in GIS 11(3): 377-397.

Jaro, M. A. (1995). "Probabilistic linkage of large public health data files." Statistics in Medicine 14.

Jones, C. B., Purves, R., Ruas, A., Sanderson, M., Sester, M., Van Kreveld, M. and Weibel, R. (2002). Spatial Information Retrieval and Geographical Ontologies: an overview of the SPIRIT project. The 25th Annual International ACM SIGIR Conference on Research and Development on Information Retrieval (SIGIR 2002), Tampere, Finland.

Levenshtein, V. I. (1965). "Binary Codes Capable of Correcting Spurious Insertions and Deletions of Ones." Probl. Inf. Transmission 1: 8-17.

Minkov, E., Wang, R. C. and Cohen, W. W. (2005). Extracting Personal Names from Email: Applying Named Entity Recognition to Informal Text. Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP), 443-450, Vancouver, Canada.

Navarro, G. (2001). "A Guided Tour to Approximate String Matching." ACM Computing Surveys 33(1): 31-88.

Patman, F. and Thompson, P. (2003). Names: a new frontier in text mining. ISI 2003, 27-38, Springer.

Pfeifer, U., Poersch, T. and Fuhr, N. (1996). "Retrieval effectiveness of proper name search methods." Information Processing and Management 326(667-679).

Souza, L. A., Davis Jr., C. A., Borges, K. A. V., Delboni, T. M. and Laender, A. H. F. (2005). The Role of Gazetteers in Geographic Knowledge Discovery on the Web. 3rd Latin American Web Congress (LAWeb 2005), 157-165, Buenos Aires, Argentina.

Winkler, W. E. (1999). The state of record linkage and current research problems, Internal Revenue Service: http://www.census.gov/srd/www/byname.html.

Zobel, J. and Dart, P. (1995). "Finding Approximate Matches in Large Lexicons." Software - Practice and Experience 25(3): 331-345.