SPARCL: An Improved Approach for Matching Sinhalese Words and Names in Record Clustering and Linkage

Gayan Prasad Hettiarachchi, Dilhari Attygalle
Department of Statistics, University of Colombo
Colombo-07, Sri Lanka

Abstract- *Quality of data residing in a database gets degraded and leads to misinterpretation due to a multitude of factors. In some cases this results in duplicate records that needs to be merged into a single entity. In doing so, one aspect requirring attention is the way in which string attributes are to be compared. Even though there are different methods in the literature that address the issue of approximate string matching, they all fall short in terms of accuracy when encountered with words from the Sinhalese language written in English. In this paper, it is intended to propose the development of an improved phonetic matching algorithm, which improved the accuracy of approximate string matching remarkably. This algorithm outperforms the phonetic matching algorithms available in the literature when applied on datasets containing Sinhalese names and words written in English. In addition, it demonstrates a computational time comparable with phonetic matching algorithms available in the literature.*

Keywords- Record Linkage, Phonetic Matching, Data Mining, Algorithm Design and Development, Clustering

1. Introduction

Quality of data residing in a database gets degraded and leads to misinterpretation of information due to a multitude of factors. Such factors vary from poor database design (update anomalies due to lack of normalization), lack of standards for recording database fields (person name and address) to typing mistakes (lexicographical errors, character transpositions). Data of such poor quality could result in many damages being caused, for example in a business application, sending wrong products and invoices to the same customer, sending products or bills to wrong addresses, inability to locate customers, etc. In such a case it is important to identify duplicates and merge them into a single entity, i.e. identify whether two entities are approximately the same. In the scientific community this process is known as record linkage [12].

A more formal definition of record linkage can be given as the task of identifying records corresponding to the same entity from one or more data sources. Real world entities of interest include individuals, families, organizations, geographic regions, etc while applications of record linkage are in areas such as marketing, customer relationship management (CRM), law enforcement, fraud detection, epidemiological studies and government administration [13].

Methods used to tackle record linkage problems fall into two broad categories: One commonly used method is deterministic models in which sets of often very complex rules or production systems are used to classify pairs of records as links (i.e. relating to the same entity). The other is the probabilistic model in which statistical or probabilistic methods are used to classify record pairs. In recent years, rapid developments of computational statistics have enabled researchers to move from classical probabilistic methods to newer and advanced approaches using maximum entropy, machine-learning techniques such as Artificial Neural Networks (ANN) and Phonetic matching [13]. Moreover, recent developments in the science of record linkage emphasize on approximate string matching more than any other aspect [3], [15].

The rationale behind focusing attention on approximate string matching is mainly driven by the fact that information about real world entities is most often represented as a collection of string attributes. The enabling technology that breathes life into duplicate identification of string attributes is phonetic matching. In order to perform this matching operation, there are different phonetic matching algorithms available in the literature. They provide a simple and time-tested mechanism for phonetic string matching. Although the simplicity of the design and implementation encourage such an approach in order to develop record linkage applications, limitations of the same are quite significant. The most highlighted drawback of phonetic matching algorithms available in the literature is its limited scope and low accuracy when encountered with words from the Sinhalese language written in English. This is probably due to the fact that phonetic matching algorithms such as Soundex, NYSIIS and Metaphone are developed for English words and it has language components, for example arrangement of vowels and consonants different from Sinhalese language. Therefore, the requirement arises for a modified version of phonetic matching algorithms to suit Sinhalese words and names. The intension of this paper is to present the development of an improved version of Soundex algorithm to suit Sinhalese names and words.

We begin by briefly describing the problem domain and the necessary background on phonetic matching algorithms. Then we describe the proposed algorithm followed by the experimental setup and the results. Finally, we bring into focus a real world application where the improved algorithm can be directly applied in order to increase performance and accuracy.

2. Problem Domain

In many matching situations, it is not possible to compare two strings exactly (character-by-character) because of

typographical errors. Dealing with typographical errors via approximate string comparison has been a major research area in computer science [1]. In record linkage, one needs to have a function that represents approximate agreement, with agreement being represented by 1 and degrees of partial agreement being represented by numbers between 0 and 1[2]. Having such methods is crucial for correct and accurate matching. For instance, in a major census application for measuring undercount, more than 25% of matches would not have been found via exact character-by-character matching [3]. Therefore, a mechanism to perform approximate string matching for datasets with typographical errors is essential in order to achieve high accuracy. The domain of the problem at hand primarily falls under the disciplines of phonetic matching, approximate string comparisons and algorithm development. However, implementing the solution requires in-depth study into several other domains of science and technology such as text retrieval, information retrieval, phonemes, etc.

Phonetic matching is used to evaluate the similarity of pronunciation of pairs of strings, independent of the characters used in their spelling. Queries to sets of strings, in particular databases of names, are often resolved by phonetic matching techniques. For example, when querying a lexicon, only the sound of a string may be known, and in addition, collections of names or words frequently contain spelling, typographical, and homonymic errors making it difficult, if not impossible, to perform one-to-one matching of strings. Thus, the requirement arises for a practical phonetic matching technique. Not only must the algorithm provide reliable judgment of similarity, but must also permit rapid evaluation of queries on a large data set: for example, lexicons of text databases can have vocabularies in excess of one million words [7].

In the literature there are several algorithms for phonetic matching, such as Soundex [6] and the more recent Phonix [4, 5]. These algorithms, which are based on the assumption that the alphabet can be partitioned into sets of sound alike characters are cheap to evaluate but do not perform well when encountered with words from the Sinhalese language. In general, the sound of a word can be described by a sequence of phonemes, which are the basic sounds available for vocalization [7]. A string of phonemes is the pronunciation of the word it represents, or for brevity, it represents sound, as distinct from the word's spelling, or string of letters. The set of phonemes is an international, language-independent standard [8]. A precise phonetic matching algorithm would regard two strings as identical if their sounds were identical, regardless of their actual spelling. It would recognize the similarity of kw and qu and of x and ecks. However, for Sinhalese, and indeed for most languages, phonemes correspond to neither individual letters nor syllables. In general it is not possible therefore, to partition a string or a sequence of letters into substrings that correspond to phonemes; nor is there any possibility of denoting phonemes in terms of sequences of individual letters [14]. In light of the above facts, it is not surprising that phonetic matching algorithms available in the literature do not perform well against Sinhalese words and names. Individual letters of Sinhalese words represented in English do not represent phonemes and many letters have very different sounds in different contexts [14]. Our aim in this research is to discover whether a modified approach to phonetic matching using Soundex might yield better performance, whether phonetic matching together with other algorithms and resources could produce accurate and better results.

3. Phonetic Matching Algorithms

Phonetic matching algorithms focus on the pronunciation of the words instead of the spellings to identify matches. Under phonetic matching, the most profound and time-tested algorithms are Soundex, NYSIIS and Phonix algorithms. A brief description of Soudex will be provided in the following section.

3.1 Soundex Algorithm

Soundex is a phonetic algorithm for indexing names by sound as pronounced in English. The goal is for names with the same pronunciation to be encoded to the same representation so that they can be matched despite minor differences in spelling [6]. Improvements to Soundex are the basis for many modern phonetic algorithms [9]. The Soundex code for a name consists of a letter followed by three numbers: the letter is the first letter of the name, and the numbers encode the remaining consonants. Similar sounding consonants share the same number, for example, B, F, P and V are all encoded as 1. Figure 1 illustrates the Soundex algorithm with respect to the different steps it contains.

- Retain the first letter of the string
- Remove all occurrences of the following letters, unless it is the first letter: a, e, h, i, o, u, w, y
- Assign numbers to the remaining letters (after the first) as follows:
  - b, f, p, v = 1
  - c, g, j, k, q, s, x, z = 2
  - d, t = 3
  - l = 4
  - m, n = 5
  - r = 6
- Remove all pairs of digits which occur beside each other from the string that resulted after the previous step
- Return the first four characters, right-padding with zeroes if there are fewer than four

Figure 1: Soundex Algorithm

4. SPARCL: The Proposed Algorithm

As described earlier in the paper the original Soundex algorithm does not perform well when encountered with words and names from Sinhalese language. This is due to the fact that Soundex, Metaphone and other algorithms were originally designed for words from English language and in addition, Individual letters of Sinhalese words represented in English do not represent phonemes and many letters have very different sounds in different contexts. Therefore, the rationale behind the development of the proposed algorithm is to make

use of the concept of phonemes and consonants from Sinhalese language to increase the accuracy of Soundex for matching Sinhalese names and words represented in English. In doing this, the Soundex algorithm given in Figure 1 had to be modified to represent consonants and phonemes from Sinhalese language. The name SPARCL stands for Sinhalese Phonetic Algorithm for Record Clustering and Linkage. Figure 2 illustrates the modified version of the Soundex algorithm. In addition, the modified algorithm was combined with a string similarity measure that takes into account the primitive number of operations that is required to transform one string to another. This addition, surprisingly, increased the accuracy of the algorithm significantly.

The new addition was facilitated by the Levenstein distance algorithm [9], which compares strings according to spelling alone with no reference to phonetic relationships [10].

One important finding is that distance measures on its own perform better than the original Soundex algorithm when applied on Sinhalese names and words. Despite the accuracy gain, distance measures alone are, however, unable to identify strings with similar sound yet dissimilar spelling such as file and phial or "Nihal" and "Neil". It is the existence of such pairs that motivates the need for a combination of distance measures and a good phonetic matching algorithm.

One way to implement distance algorithms is to measure the closeness in terms of the number of primitive operations necessary to convert the string into an exact match. To be more precise, let P be a pattern string and T a text string over the same alphabet. The Levenstein distance between P and T is the smallest number of changes sufficient to transform a substring of T into P, where the changes may be:

Substitution - two corresponding characters may differ: Rathnapure ⟶ Rathnapura.
Insertion - we may add a character to T that is in P:
Ratnapura ⟶ Rathnapura.
Deletion - we may delete from T a character that is not in P:
Ratthnapura ⟶ Rathnapura.

The algorithm is implemented using a dynamic programming approach that calculates the number of edits D between every possible left-sided substring of each of the two words a and b. $D(a_i, b_j)$, for example, is the edit distance between the first i letters of the word a and the first j letters of the word b. The dynamic programming calculation is recursive, where C_I, C_M, and C_D, are the costs of insertion, substitution, and deletion respectively. In the simplest case, the costs of insertion, deletion, and substitution are all unit costs, and the cost of a match is zero. That is, $C_I(x) = C_D(x) = 1$ for all x and $C_M(x,y) = 0$ if x = y, 1 otherwise.

$$D(a_i,b_j) = \min \begin{pmatrix} D(a_i,b_{j-1}) + C_I(b_i) \\ D(a_{i-1},b_{j-1}) + C_M(a_i,b_i) \\ D(a_{i-1},b_j) + C_D(a_i) \end{pmatrix} \quad (1)$$

Results of the experiments carried out on the combined approach will be presented in the next section. It is worthwhile discussing even though it has not been implemented yet, another possible extension that could further improve the accuracy of the proposed algorithm. It is clear that there is no exact algorithm for deriving the likely sound of a string. However, a statistical approach can be adopted to realize this purpose. In the literature there are table books that provide phonemes, phoneme strings and spellings that include the corresponding sounds, but these sources do not provide statistics of the likelihood of correspondence [11]. An alternative approach is to use a software dictionary that provides the spellings and pronunciation of words. The sounds or the pronunciation given in the dictionary provides an alternative approach to phonetic matching. The purpose of phonetic matching can be directly substituted by this method. Adopting this approach, together with distance measures can provide accuracy comparable to the modified approach.

- Retain the first letter of the string
- Remove all occurrences of the following letters, unless it is the first letter: a, e, h, i, o, u, w, y
- Assign numbers to the remaining letters (after the first) as follows:
  - b, f, p, v = 1
  - c, g, j, k, z = 2
  - d -> j if in -dge-, -dgy- or -dgi-
  - q -> k
  - s -> x (sh) if before "h" or in -sio- or -sia-
  - s otherwise
  - s -> x
  - t -> x (sh) if -tia- or -tio-
  - t -> o (th) if before "h"
  - t -> t otherwise
  - l = 4
  - m, n = 5
  - r = 6
  - x -> ks
  - z -> s
- Remove all pairs of digits which occur beside each other from the string that resulted after the previous step
- Return the first four characters, right-padding with zeroes if there are fewer than four

Figure 2: Proposed SPARCL Algorithm

5. Implementation

The implementation of the SPARCL algorithm was carried out in C# programming language under Microsoft Visual Stdio.NET development environment.

The code of the algorithm is organized to optimize the matching process by avoiding unnecessary execution of loops, recursion and redundant comparison of strings. In addition, the code conforms to a comprehensive coding standard enforcing best practices and avoiding pitfalls. The Graphical User Interface (GUI) provides functionality to select any word list stored at a particular location. In addition, functionality is provided to easily apply the SPARCL algorithm on a lexicon and test the accuracy and computational time of the process. The outputs provide the similar pairs of words and names as identified by the algorithm. Furthermore, Soundex algorithm and the Levenstein distance algorithm can also be applied on the specified lexicon in order to carry out a performance

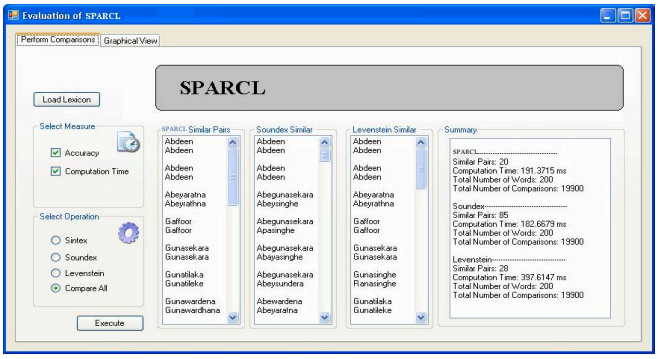

Figure 3: Outputs Obtained by the Algorithms

comparison between the three algorithms. Figure 3 illustrates the outputs obtained by executing the three algorithms on a dataset containing 200 Sinhalese names. Further explanation of the outputs will be provided in section 6.

6. Empirical Evaluation

We now describe our empirical evaluation of the SPARCL algorithms' accuracy and computation time. The datasets that have been used for evaluating SPARCL are described below.

6.1. Datasets

For measuring the accuracy of the SPARCL algorithm we used a dataset containing 200 Sri Lankan surnames. All entries were distinct except for 20 similar surnames with typographical errors, which were deliberately incorporated into the dataset. In addition, a different dataset with 10000 entries were created to compare the computational time of the proposed SPARCL algorithm against the original Soundex algorithm. In addition to comparing the improved algorithm with the original Soundex algorithm, it was compared with the Levenstein distance algorithm.

6.2. Experimental Setup

The accuracy of the SPARCL algorithm was compared with the original Soundex algorithm as well as the Levenstein distance algorithm. This was accomplished using the dataset containing 200 Sri Lankan surnames. Each word in the dataset was compared with the rest of the words only once and any similar pair of words was counted as a match.

Similarly, for testing the computation time of the SPARCL algorithm a dataset of 10000 words was utilized. Each time the SPARCL algorithm was executed on a subset of the dataset. Similarly, the original Soundex algorithm and the Levenstein distance algorithm was executed on the same subset and the computation time or the running time of the algorithms were measured in milliseconds. The obtained results will be discussed in the following sections.

6.3. Accuracy

The SPARCL algorithm produced the same set of similar word pairs that are actually present in the dataset. As described above under section 6.1, we deliberately incorporated 20 pairs of similar words with typographical mistakes. The SPARCL algorithm exactly produced the same 20 pairs of words as matches. Original Soundex algorithm on the other hand, produced 68 incorrect pairs in addition to 17 correct pairs. Similarly, the Levenstein algorithm produced 12 incorrect pairs in addition, to 16 correct pairs. Therefore, SPARCL algorithm demonstrates an accuracy of 100% and an error rate of 0%. (We note that this 100% accuracy was obtained for a dataset with 200 records). Soundex algorithm demonstrates an accuracy of 85%. However, this high accuracy rate is highly compromised by the fact that it produces another additional 68 incorrect matches. Similarly, Levenstein distance algorithm demonstrates an accuracy of 80% with additional 12 incorrect matches. It is clear from the results that Levenstein algorithm alone performs better than the original Soundex algorithm when encountered with Sinhalese words. However, SPARCL, which combines a modified version of Soundex and Levenstein distance algorithms to suit Sinhalese names and words, outperforms the other two methods by quite a margin.

6.4. Computational Time

We measured the computation time for all three algorithms separately using a dataset of 10000 names. The algorithms were run on Windows XP using a 1.66 GHz Intel Centrino Duo machine with 512 MB of RAM. Different subsets of the dataset were created ranging from 200 entries to 10000 entries. For all subsets, computation times were within an order of magnitude of each other for all three algorithms. Original Soundex algorithm is faster than both the proposed algorithm and the Levenstein distance algorithm. However, the proposed approach and the original Soundex produced comparable results with respect to computational time. Even though, Soundex demonstrates a small performance edge over the proposed algorithm, it is highly compromised by the huge reduction in accuracy when encountered with words from Sinhalese language. The computational times for each of the subsets are given in Table 1. Figure 4 illustrates a graphical representation of the computational times of the three algorithms in the same chart.

One area where duplication is quite evident and record linkage proves to be quite useful is with regard to newspaper articles. We consider such articles related to human right violations. Articles related to the same incident are published in various newspapers on different days in various ways. Also the same incident is reported and discussed in the same newspaper continuously for several days. This leads to the problem of duplication which can in turn lead to other large scale problems at the national and international level. This type of duplication can reveal an incorrect image about the country's situation to local as well as international communities.

Such duplication can lead to an over estimation of the violations taken place in a country and could be even more severe when there are ethnic issues or a civil war going on in a country like Sri Lanka. Record linkage can help alleviate these problems to a great extent. A dataset of human right violations would generally consist of attributes such as victim name, location, incident type, perpetrator, etc. These attributes are often represented as character strings. In order to perform

Table 1: Comparison of Computational Times

Number of Words	Computational time in milliseconds		
	SPARCL algorithm	Original Soundex algorithm	Levenstein distance algorithm
200	105.8	94.6	127.8
500	613.3	587.1	807.1
1000	2506.8	2371.5	3241
2000	9634.5	9570	13255.5
3000	21668.5	21560	29942.8
4000	39001.9	38866.5	54619.8
5000	61864.6	61384.2	89290.2
6000	89166.4	87390.9	129449.9
7000	121870	121321.9	172661.9
8000	160682	159197.5	224700
9000	205752.8	207589.3	292965.4
10000	253989.5	255694.7	362773.3

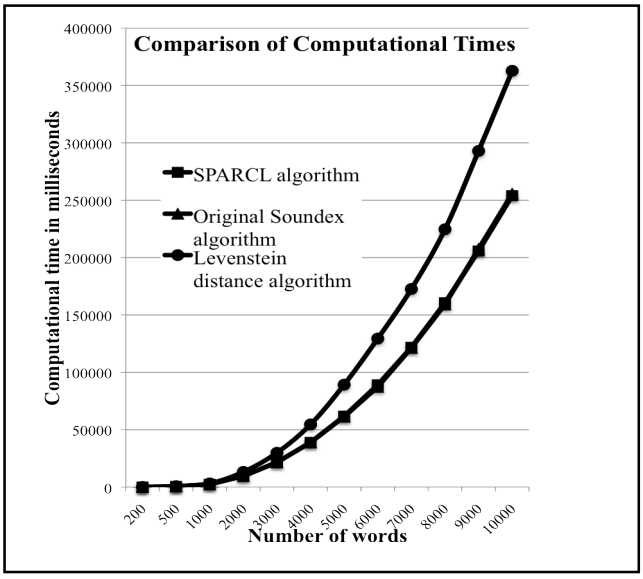

Figure 4: Comparison of Computational Times: Both SPARCL and Soundex algorithms show comparable results with respect to computational time. Therefore, lines corresponding to these two algorithms are almost overlapped in Figure 4.

record linkage on these attributes we can make use of phonetic matching. Since these attributes contain names and words from Sinhalese language, the original Soundex algorithm as mentioned earlier is unable to provide the required level of accuracy.

However, according to results given in section 6, the proposed SPARCL algorithm can provide the required foundation to achieve high level of accuracy. In addition, the use of SPARCL algorithm for this purpose will not result in a huge performance loss due to the fact that it demonstrates a comparable computation time with the original Soundex algorithm.

Apart from the aforementioned purpose, the SPARCL algorithm can also be used in the development of a generic framework for record classification and linkage. The SPARCL algorithm will lie in the center of the framework providing the backbone for duplicate identification of string attributes. In addition, the framework can provide additional functionality such as clustering, blocking, weighting attributes, selection of blocking variables, prediction of missing attribute values, etc required by a general record linkage program. The development of the framework is under working progress.

7. Conclusion

This paper proposes a modified phonetic matching algorithm as an alternative to both Soundex algorithm and the Levenstein distance algorithm for comparing names and words from Sinhalese language written in English. Experiments show that SPARCL produce better results in terms of accuracy than the other two approaches. In terms of performance, both SPARCL algorithm and Soundex algorithm show similar computational times on a number of datasets. However, Levenstein distance algorithm shows very low performance relative to the other two approaches.

Directions for future research include refining the algorithm to make use of the pronunciations provided in an online or offline dictionary to perform phonetic matching. In addition, the proposed SPARCL algorithm can be applied to a myriad of real world problems including the development of a framework for record classification and linkage.

References

[1] Hall, P. A. V., and Dowling, G. R. (1980), "Approximate String Comparison," Computing Surveys, 12, 381-402.

[2] Winkler, W. E. (1995), "Matching and Record Linkage," in B. G. Cox et al. (ed.) Business Survey Methods, New York: J. Wiley, 355-384.

Verykios (2007), "Duplicate Record Detection: A Survey" , IEEE Transactions on Knowledge and Data Engineering 19 (1): pp. 1–16

[4] T.N. Gadd. (1988), `Fisching fore werds': Phonetic retrieval of written text in information systems. Program: automated library and information systems, 22(3):222-237.

[5] T.N. Gadd. (1990), PHONIX: The algorithm. Program: automated library and information systems, 24(4):363

[6] P.A.V. Hall, G.R. Dowling (1980), Approximate string matching. Computing Surveys, 12(4):381{402, 1980.

[7] A.C. Gimson and A. Cruttenden (1994), Gimson's Pronounciation of English. Edward Arnold, London, _fth edition.

[8] (2007), The principles of the International Phonetic Association. Phonetics Department, University College, London, UK.

[9] J. Zobel and P. Dart. (1995), Finding approximate matches in large lexicons. Software Practice and Experience, 25(3):331.

[10] K. Kukich. (1992), Techniques for automatically correcting words in text. Computing Surveys, 24(4):377.

[11] Nielsen, Sandro (2008), "The effect of lexicographical information costs on dictionary making and use", in Lexikos (AFRILEX-reeks/series 18), pp.170–189

[12] G. Salton and M.J. McGill. Introduction to Modern Information Retrieval. McGraw-Hill, New York, 1983.

[13] Alvey, W. and Jamerson, B. (eds.) (1997), Record Linkage Techniques -- 1997 (Proceedings of An International Record Linkage Workshop and Exposition, March 20-21,1997, in Arlington VA), Washington, DC: Federal Committee on Statistical Methodology.

[14] Dulip Herath, Kumudu Gamage, Anuradha Malalasekara, Research Report on Sinhala Lexicon. [Online]. Available: http://www.panl10n.net/english/final%20reports/pdf%20files/Sri%20Lanka/SRI01.pdf
[Accessed: Apr, 29, 2008]

[15] Rohan Baxtor, Peter Christen, Tim Churches (2003), A Comparison of Fast Blocking Methods for Record Linkage, Workshop on Data Cleaning, Record Linkage and Object Consolidation, 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington DC