Particle Swarm Optimization in Transliteration

Dr. Pothula Sujatha

Assistant Professor, Department of Computer science
School of Engineering & Technology, Pondicherry University
Pondicherry, India.

Abstract— **Transliteration is the process of transforming a word written in a source language into a word in a target language without the aid of a resource like a bilingual dictionary. This process generates the target language word for a given source language word, but need to find the similarity between source and target words. That is, in order to check how far the generated target word is right equivalent an edit distance calculation is needed between source and target languages words. Presently there was no automated process for finding edit cost between source and target languages words. This work proposes a new Particle Swarm Optimization (PSO) algorithm which is used in the transliteration algorithm process for finding optimal cost between source and target words.**

Keywords—**Swarm intelligence, particle swarm optimization, transliteration, grapheme, phoneme, hybrid**

I. INTRODUCTION

Transliteration is the process of transforming a word written in a source language into a word in a target language without the aid of a resource like a bilingual dictionary. It refers to expressing a word in one language using the orthography of another language. Orthography means the art or study of correct spelling according to established usage. Transliteration can be classified into two directions. Given a pair (s, t), where s is the source word in source language and t is the transliterated word in target language. Two types of transliteration are available in the literature: forward and backward translation. Forward transliteration is the process of converting s into t. Backward or back transliteration is the process to correctly find or generate s for a given t. Backward transliteration is more challenging than forward transliteration. This paper is adopting forward transliteration.

The major techniques for transliteration can be classified into three categories: grapheme-based transliteration model, phoneme-based transliteration model, and grapheme- and phoneme-based transliteration model [5].

Grapheme refers to the basic unit of written language (or smallest contrastive units). For example, English has 26 graphemes or letters (21 consonants and 5 vowels), Hindi has 46 letters (33 consonants and 13 vowels), Tamil has 30 letters (18 consonants and 12 vowels), and Telugu has 56 letters (40 consonants and 16 vowels). Grapheme-based transliteration (spelling) is referred to as the direct method because it directly transforms source language graphemes into target language graphemes without any phonetic knowledge of the source language words. Here transliteration is identified by mapping the source language names to their equivalent names in a target language and generating them.

Phoneme refers to the simplest significant unit of sound or the smallest contrastive units of a spoken language. In Phoneme-based transliteration pronunciation rather than spelling of the original string is considered as a basis for transliteration. Phoneme based transliteration is referred as a pivot method because it uses source language phonemes as a pivot, when it produces target language graphemes from source language graphemes. It usually needs two steps:

- Produce source language phonemes from source language graphemes.
- Produce target language graphemes from source phonemes.

These two steps are explicit if the transliteration system produces target language transliterations after producing the pronunciations of the source language words; they are implicit if the system uses phonemes implicitly in the transliteration stage and explicitly in the learning stage [6]. ARPAbet symbols are used to represent source phonemes. ARPAbet is one of the methods used for coding source phonemes into ASCII characters [7]. Grapheme-based and phoneme-based transliteration is referred to as hybrid transliteration. It makes use of both source language graphemes and source language phonemes, when producing target language transliterations. Here after, a source language grapheme is a source grapheme, a source language phoneme is a source phoneme, and a target language grapheme is a target grapheme.

In each model, transliteration of a source language to target language is an interesting and challenging task. This paper proposes transliteration algorithm and PSO algorithm. Grapheme based transliteration is used for transliterating source name to target name and edit distance between source and target is calculated using the proposed similarity PSO algorithm.

The organization of the paper is as follows. Section II, describes previous studies on PSO algorithm and transliteration technique. Section III, explains the proposed transliteration algorithm along with PSO similarity algorithm. Conclusion and future scope of the paper is given in section IV.

II. RELATED WORK

A lot of research is going on in SI particularly PSO. The biologic principles of SI are explained in [8]. Extension of PSO called Geometric Particle Swarm Optimization (GPSO) is described in [2]. This GPSO can be applied to both continuous and combinatorial spaces. Tree based edit distance using PSO is explained in [9]. The model is efficient and more transparent as probabilistic approaches as well as less complexity.

An n-gram based statistical transliteration model for English to Arabic names was described in [10]. It presents a simple statistical technique, which does not require any heuristics or linguistic knowledge of either language. It is specified that transliteration either of Out Of Vocabulary (OOV) named entities or of all OOV words is an effective approach for CLIR. A decision tree based transliteration model [11] is a language independent methodology for English to Korean transliteration and back transliteration. It is composed of character alignment and decision tree learning. Transliteration rules and back transliteration rules are induced for each English alphabet and each Korean alphabet. A maximum entropy based model [12] is an automatic transliteration model from English to Japanese words and it successfully transliterates an English word not registered in any bilingual or pronunciation dictionaries by converting each partial letters in the English word into Japanese characters. A new substring based transliteration method based on phrase-based models of machine translation was described in [13]. Substring based transliteration method is applied on Arabic to English words. A rule based model for English to Korean transliteration using pronunciation and context rules is described in [14]. It uses phonetic information such as phoneme and its context as well as orthography of English language as the basis for transliteration. A machine-learned phonetic similarity model [15] is a backward transliteration model and provides learning algorithm to automatically acquire phonetic similarities from a corpus. Given a transliterated word, similarity based model compares the list of source candidate words and the one with highest similarity will be chosen as the original word. Oh and Choi [16] proposed a model for improving machine transliteration using an ensemble of three different transliteration models (grapheme, phoneme and both) for English to Korean and English to Japanese languages. Bilac and Tanaka [17] proposed a new hybrid back transliteration system for Japanese, which contains segmentation, phoneme-based and grapheme-based transliteration modules. Context, transliteration similarity mechanism to align English-Hindi texts at the sentence and word level in parallel corpora was proposed by [18]. This is based on a grapheme-based model. It describes a simple sentence length approach to perform sentence alignment and multi feature approach to perform word alignment. Punjabi machine transliteration was described by [19]. A word origin based approach for splitting Indian and foreign origin words and transliterating them based on their phoneme equivalents was shown by [20]. The given transliteration mechanism is applicable for Indian languages and shown that word origin is an important factor in achieving higher accuracy in transliteration. A phrase based (grapheme-based) statistical machine transliteration of named entities from English to Hindi using a small set of training and development data was presented by [21].

III. PROPOSED ALGORITHMS

Transliteration is the process of transforming a word written in a source language into a word in a target language without the aid of a resource like a bilingual dictionary [22]. Transliteration algorithm from source language name to target language is shown in Fig 1. and notations used in the proposed algorithms are given in Table 1. It uses grapheme based model for transliterating source to target name. The proposed transliteration algorithm contains character level alignments between source and target languages. i.e. for a given Source Name (SN), it divides into individual characters (S_1^(n)) and perform the level wise alignments using intermediate scheme. The roman scheme is used as intermediate scheme for transliteration. Mapping is done with either one-to-many and one-to-one configuration or many-to-one and one-to-one configuration [22]. The selection of the scheme depends on the source and target languages. Using roman scheme individual target characters (T_1^(m)) are generated and finally Target Name (TN) is generated. After generating TN edit distance is calculated between SN and TN for finding how far generated transliteration is right equivalent. Normally similarity measures are used to calculate edit distance between source and target languages but the process is manual.

Transliteration Algorithm:

```
Transliteration ($SN$)

Start
    Initialize $n$
        For $i = 1$ to $n$ do
            Divide $SN$ into $S_1^n$
            Character level alignments // n-gram alignment
            Generate $T_1^m$ using roman scheme
            Generate $TN$
        End For
    $\varepsilon$ =PSOEditCost ($SN$,$TN$)
        If $\varepsilon = 0$ then
            Exact equivalent is $TN$
        Else
            Exact equivalent is not $TN$
        End If
    Return $TN$
End
```

Fig.1If necessary, the images can be extended both columns Transliteration Algorithm

The main goal is to automate the similarity measure between source and target languages. For that proposed a PSO algorithm which calculates optimal edit cost between SN and TN. The proposed PSO edit distance algorithm is shown in Fig 2. and the notations used in the PSO algorithm are given in Table 1. First it generates the random swarm particles (P_1^(k)). For each particle defined three cost operations. These operations include insertion (α), deletion (β) and substitution (γ). Insertion operation includes insertion of character, deletion operation includes deletion of the character and substitution includes substituting a character. These operations are applied for finding the similarity measure between source and target names. For each particle set randomly their position (〖Pos〗 _k^l) and relative velocities (V_k^l). After setting their positions, for each particle obtain the fitness function value based on the following objective function or fitness function (OF).

$$OF = D(S_1^n, T_1^m)$$
$$= min\,\{\alpha, \beta, \gamma\}$$
$$= \left(\begin{array}{l} min\,\{D(n-1,m)+1,\ D(n,m-1)+1,\ D(n-1,m-1)\ +\ \Phi\} \\ \qquad\qquad Where \left\{ \begin{array}{ll} \Phi = & S_x = T_y \\ 1 & Else \end{array} \right\} \end{array} \right)$$

The objective function is based on the minimum number of edit operations (between source and target language names. Each particle in the swarm has local best fitness value . Each time the local best fitness value is compared with the new position fitness function value ( ). If is better than that of then the new position fitness function is assigned as a local best fitness value. From the of all individual particles set the global position as To attain update velocity (and position ( ) of individual particles. The updation can be performed using the following equations.

PSO Algorithm:

```
PSOEditCOST (          )
Start
        Generate Swarm of particles
        For each particle       do
                // Define 3 cost operations
                          ⎧ → Insertion        ⎫
    =          → Deletion ⎨                    ⎬
  → Substitution          ⎩                    ⎭
                   Set random position
                   Set random velocity
            End For
            DO
            For each particle's position            do
            // obtain fitness function value
   =
=

        =   ⎛          Where    ⎧ =           ⎫⎞
            ⎝                   ⎩ 1    Else   ⎭⎠

  If fitness (      ) is better than fitness (        ) then

            End If
                   Set best of          as
            For each particle     do
            //Update velocity

            //Update position

            End For
Until (Maximum number of iterations is reached probably set as 10)
            Return (     )
End
```

Fig. 2PSO Edit Calculation

TABLE I
ALGORITHM NOTATIONS

Notation	Description
	Source language name
	Target language name
	Source name length
	Target name length
= { , … }	Individual source characters
= { , … }	Individual target characters
	Edit distance
= { , … }	Random swarm of particles
	Cost operations
	{ 0,1}
	Objective function or Fitness function
	Fitness function value
	Best local position of the particle based on fitness (solution)
	Global position from the fitness of all individual neighborhood particles
	Velocity of the particle's at iteration
	Position of the particle's at iteration
	Random variables drawn from uniform distribution in the range [0,1]
	Two acceleration constants or Learning factors (usually = = 2)
	Weighting function (usually selected less than 1 for global exploration)
	Optimal Cost
	Constants

IV. CONCLUSIONS

PSO is a population based stochastic global optimization algorithm. A novel PSO algorithm is proposed for calculating optimal edit cost between source and target language name. This PSO algorithm is used in the transliteration systems which follows grapheme based character level alignment process for transliterating source to target name. The proposed PSO algorithm can automate the process for calculating optimal cost. This current work can be extended to applying Ant Colony Optimization (ACO) algorithm in the transliteration process for reducing transliteration errors.

REFERENCES

[1] E. Bonabeau, M. Dorego and G. Theraulaz, "Swarm Intelligence: From natural to artificial systems" *Bio-Inspired Computing*, 2003.

[2] Julian Togelius , Renzo De Nardi , Alberto Moraglio, "Geometric PSO + GP = Particle Swarm Programming," *In: Proceedings of CEC 2008*, pp. 3594–3600.

[3] M. Clerc, "Discrete particle swarm optimization, illustrated by the traveling salesman problem," *New Optimization Techniques in Engineering, Springer*, 2004, pp. 219–239.

[4] J. Kennedy and R. C. Eberhart, A discrete binary version of the particle swarm algorithm, *IEEE Transactions on Systems, Man, andCybernetics* ***5*** *(1997)*, 4104–4108.

[5] J. H. Oh and K. S. Choi, “An ensemble of transliteration models for information retrieval,” *Information Processing and Management: an International Journal*, v.42 n.4, pp. 980-1002, July 2006.

[6] S. Bilac and H. Tanaka, “Improving back-transliteration by combining information sources,” *In Proceedings of IJC-NLP*, pp. 542–547, 2003.

[7] D. Jurafskyand J. H. Martin, “Speech and Language processing: An introduction to natural language processing,” *Computational Linguistics and Speech Recognition*, 2007.

[8] S. Garnier, J. Gautrais and G. Theraulaz, “The biological principles of swarm intelligence,” *Swarm Intelligence*, pp. 3-31, 2007.

[9] Y. Mehdad, “Automatic cost estimation for tree edit distance using particle swarm optimization,” *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 289–292, 2009.

[10] N. A. Jaleel and L. S. Larkey, “Statistical transliteration for english-arabic cross language information retrieval,” *In Proceedings of the twelfth international conference on Information and knowledge management*, November 03-08, 2003, New Orleans, LA, USA.

[11] B. J. Kang and K. S. Choi, “Automatic transliteration and back-transliteration by decision tree learning,” *In: Proc. Of the Second International Conferenceon Language Resources and Evaluation*, 2000.

[12] W. H. Lin and H. H. Chen, “Backward machine transliteration by learning phonetic similarity,” *In: Proc. of the Sixth Conference on Natural Language Learning*, pp. 139–145, 2002.

[13] T. Sherif and G. Kondrak,”Substring based transliteration,” In proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pp. 944-951, 2007.

[14] J. H. Oh and K. S. Choi, “An English-Korean transliteration model using pronunciation and contextual rules,” *In: Proc. Of the 19th International Conference on ComputationalLinguistics*, pp. 758–764, 2002.

[15] [15] W. H. Lin and H. H. Chen, “Backward machine transliteration by learning phonetic similarity,” *In: Proc. Of the Sixth Conference on Natural Language Learning*, pp. 139–145, 2002.

[16] J. H. Oh and K. S. Choi, “An ensemble of transliteration models for information retrieval,” *Information Processing and Management: an International Journal*, v.42 n.4, pp. 980-1002, July 2006.

[17] S. Bilac and H. Tanaka, “A hybrid back-transliteration system for Japanese,” *In: Proc. Of the 20th International Conference on Computational Linguistics* (COLING 2004), pp. 597–603, 2004.

[18] N. Aswaniand and R. Gaizauskas, “A hybrid approach to align sentences and words in English-Hindi parallel corpora,” *In Proceedings of the ACL Workshop on Building and Exploiting Parallel Texts*, 2005.

[19] M. G. A. Malik. “Punjabi machine transliteration,” In Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL, pp. 1137–1144, 2006.

[20] H. Surana and A. K. Singh, ” A more discerning and adaptable multilingual transliteration mechanism for Indian languages,” *In Proceedings of International Joint Conference on Natural Language Processing (IJCNLP), 2008*, Hyderabad, India.

[21] T. Rama and K. Gali, “Modeling machine transliteration as a phrase based statistical machine translation problem,” *In proceedings of the Named Entities Workshop, ACL–IJCNLP’09*, pp. 124-127, August 2009.

[22] VasiNarasimhulu, P. Sujatha, P. Dhavachelvan and M. S. SaleemBasha, “Enhanced Named Entity Transliteration Model Using Machine Learning Algorithm”, International journal of Advancements in Computing Technology Volume 2, Number 3, August, 2010.