

English to Tamil Transliteration using WEKA

Vijaya MS, Ajith VP, Shivapratap G, and Soman KP

CEN, Amrita University, Coimbatore, India

Email: {ms_vijaya, vp_ajith, g_shivapratap, kp_soman}@ettimadai.amrita.edu

Abstract—Machine transliteration has gained prime importance as a supporting tool for Machine translation and cross language information retrieval especially when proper names and technical terms are involved. The performance of machine translation and cross-language information retrieval depends extremely on accurate transliteration of named entities. Hence the transliteration model must aim to preserve the phonetic structure of words as closely as possible. In this paper, the transliteration problem is modeled as classification problem and trained using C4.5 decision tree classifier, in WEKA Environment. The training was implemented with features extracted from a parallel corpus. This technique was demonstrated for English to Tamil Transliteration and achieved exact Tamil transliterations for 84.82% of English names. Possible equivalent transliterations were also generated by the model. It is found that the transliteration accuracy is increased when the top five ranked transliterations were considered.

Index Terms—Transliteration, Sequence labeling, alignment, decision tree, WEKA.

I. INTRODUCTION

One of the major issues needed to be considered by translators is translating named entities such as person, location, organization names and technical terms. During machine translation, the most proper way to handle named entities is to transliterate them into the target language by closely preserving their phonetic structure. Also proper noun processing plays an important role in query translation during cross-language information retrieval, where the query is specified in a language different from that of the retrieved documents. Thus machine transliteration is an important and challenging task in machine translation, cross-language information retrieval, multi lingual text and speech processing.

During the last decade, four kinds of transliteration models have been developed. These are grapheme-based model, phoneme-based model, hybrid model, and correspondence-based model. These models are classified based on the units to be transliterated such as graphemes, phonemes, both graphemes and phonemes, and the correspondence between graphemes and phonemes.

This paper presents a transliteration model where the transliteration problem is modeled using classification technique and implemented using WEKA j48 decision tree classifier. The transliteration model is experimented on English to Tamil transliteration and its performance is evaluated.

II. TRANSLITERATION PROBLEM AND CLASSIFICATION

Transliteration is a process that accepts a character string in source language as input and generates a

character string in the target language as output. The transliteration process mainly involves the segmentation of the source string into transliteration units and relating the source language transliteration units with units in the target language by resolving different combinations of alignments and unit mappings. The learning of the transliteration model takes place based on the alignments[1][2].

III. CORPUS DEVELOPMENT AND FEATURE GENERATION

The performance of the transliteration model hugely depends on the data with which it is trained. Hence it is important to have a large corpus with many examples that highlight the intricacies of the language. A bilingual corpus consisting of 30,000 Indian person names and 30,000 Indian place names was created. This corpus was used in training the transliteration model. The source language considered here is English and the target language is Tamil.

The valid target language n-gram (y_i) for a source language n-gram (x_i) in the given source language input word is decided by considering the source language context features such as source language n-gram (x_i), two left context n-grams (x_{i-2} , x_{i-1}) and two right context n-grams (x_{i+1} , x_{i+2}). A centered window of five tokens have been considered, from which basic and n gram patterns are evaluated to form binary features. In addition to context features, the features specifying the start and end of words, are extracted and encoded as binary features. Thus a binary feature vector of size 402 is created for each x_i . The feature patterns corresponding to each source language n-gram x_i are generated to form a multi class training set. Such training datasets developed with binary features are used for training.

The corpus needs to be preprocessed before training the transliteration model. This involves conversion of English words to Romanized Tamil counterparts, segmentation of these Romanized Tamil words, and alignment[1][2]. During alignment, the corresponding transliteration units in English and Tamil words are aligned if the number of units in both English and Tamil words are equal.

A. Mapping Analysis

An English n-gram may be mapped into one or more Tamil n-grams. A dictionary is extracted from the training corpus with all possible mapping class labels for each English n-gram with mapping label frequency. The dictionary is referred during the training and the

prediction of class label for a given source language n-gram. The mapping rules for all the English n-grams are obtained using WEKA j48 classifier. The mapping label with less frequency is not considered for generating rules. The two left context features for an English n-gram xi are denoted as lw-1 and lw-2 and the two right context features as rw-1 and rw-2.

VI. TRANSLITERATION USING WEKA

Since every English n-gram is mapped into more than one class labels, predicting a class label for each English n-gram in the name to be transliterated becomes a multi class classification problem. The Waikato Environment for Knowledge Analysis (WEKA) is a comprehensive suite of Java class libraries that provides implementations of numerous classification and prediction algorithms [3]. C4.5 Decision tree classifier of WEKA which is a supervised machine learning algorithm has been used here for training the classification model and to generate rules.

Decision tree classification generates the output as a binary tree like structure called a decision tree, in which each branch node represents a choice between a number of alternatives, and each leaf node represents a classification or decision. A decision tree model contains rules to predict the target variable [4]. This algorithm scales well, even where there are varying numbers of training examples and considerable numbers of attributes in large databases.

A. Transliteration System

The transliteration process consists of four phases: Pre-processing phase, feature extraction, training and transliteration phase.

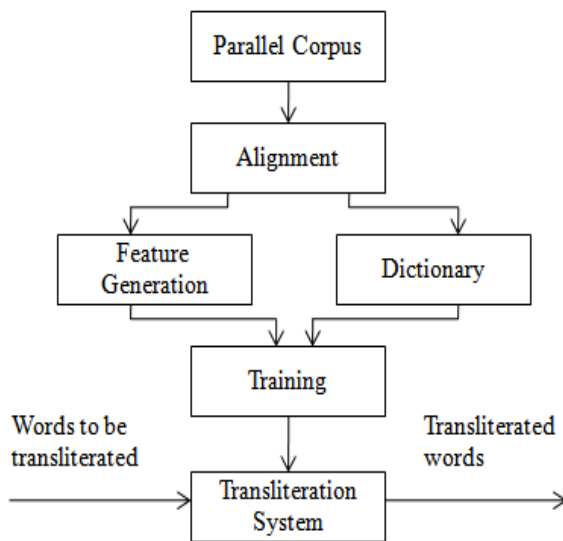


Fig. 1. Transliteration System

During the preprocessing phase, the target language names are Romanized and the source language names are segmented and aligned with the corresponding segmented target language names.

The aligned corpus was identified to consist of 80 unique English n-grams and 161 unique labels. A dictionary containing English n-grams with their mapping labels and the mapping label frequency is generated. The feature patterns are extracted from the aligned corpus as described in section 3. For each source language n-gram, a multi class training dataset is created and trained using WEKA j48 decision tree classifier. The trained decision tree model contains a set of rules for predicting class label. When predicting the class labels for a given sequence of source language n-grams, the most probable class label for each English n-gram is predicted using the respective trained model. The trained models are incorporated into English to Tamil Transliteration system.

During Transliteration phase, each word that is to be transliterated is first segmented into a sequence of English n-grams. The feature vector for each English n-gram is extracted. The trained model, corresponding to each n-gram in English word is used for prediction. The sequence of predicted class labels will form a transliterated word for the given English word.

B. Results and Evaluation

The transliteration system was tested with a data set of 1000 English names. The transliteration system based on decision tree classifier produced exact Tamil transliteration for English words with an accuracy of 84.82%. The misclassifications by the model can be attributed to the delicate characteristics of the Tamil language while transliterating vowels and the consonants 'r', 'l', 'n', 'd', 't'. Several possible transliterations are made based on the second best class label for the vowels and the above mentioned consonants. Considering only the top 2 results, 87.19% of accuracy is obtained. If the top 5 results are considered, the accuracy is increased to 93.15%. The accuracy can be increased when the number of possible transliterations is increased. The number of transliterations can be increased by considering the next best class labels.

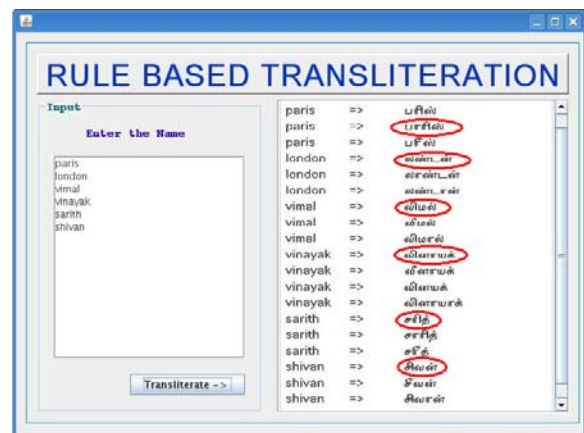


Fig. 2. Sample Screenshot

TABLE VI
TRANSLITERATION ACCURACY

Output	Accuracy
Top1	84.82%
Top 2	87.19%
Top 3	88.4%
Top 4	91.57%
Top 5	93.15%

The results of the system was compared with the Google Indic Transliteration (<http://www.google.co.in/transliterate/indic>) results and observed that the overall accuracy for Google when transliterating Indian person names and place names was 68 % when considering the first choice.

VII. CONCLUSION

A transliteration model based on multi class classification approach for English to Tamil transliteration was demonstrated. In this model, j48 decision tree classifier of WEKA was used for classification. The feature patterns were extracted from the parallel corpus consisting of 30,000 person names and 30,000 place names and used it to train the transliteration model. The accuracy of the model was tested with 1000 English names that were out of corpus. The transliteration model produced an exact transliteration in Tamil from English words with an accuracy of 84.82%. The accuracy can be increased by considering the next best output from the classifier. This can also be used to generate all possible transliterations for a given English word.

The transliteration model also performed better when compared to Google Indic transliteration. But the fact is that the Google system is designed for general transliteration whereas the model presented here is trained exclusively for Indian names and places. It is concluded that this transliteration model is applicable for the languages which have the same alpha-phonetic sequence in both source and target languages.

REFERENCES

- [1] Vijaya MS, Loganathan R, Shivapratap Gopakumar, Ajith VP, Soman KP "English To Tamil Transliteration Using Sequence Labelling Approach", *International Conference on Asian Language Processing, Thailand*, November 2008
- [2] Vijaya MS, Dhanalakshmi V, Shivapratap G, Ajith VP, Soman KP, "Sequence Labeling Approach for English to Tamil Transliteration using Memory based Learning", *Proceedings of 6th International Conference on Natural Language Processing ICON*, Pune, December 2008
- [3] Ian H. Witten, Eibe Frank, Len Trigg, Mark Hall, Geoffrey Holmes, Sally Jo Cunningham. 1999. *Weka: Practical Machine Learning Tools and Techniques with Java*
- [4] Ian H. Witten, Eibe Frank. 2005. *Data Mining – Practical Machine Learning Tools and Techniques*. Second Edition. Elsevier.
- [5] *Implementations*. Working Paper 99/11, Department of Computer Science, The University of Waikato, Hamilton.
- [6] Arababi Mansur, Scott M. Fischthal, Vincent C. Cheng, Elizabeth Bar. 1994. *Algorithms for Arabic name transliteration*. IBM Journal of Research and Development.
- [7] Knight Kevin and Graehl Jonathan. 1997 *Machine transliteration*. In proceedings of the 35th Annual Meeting of the Association for Computational Linguistics. 128-135.
- [8] Al-Onaizan Y, Knight K. 2002 *Machine translations of names in Arabic Text*. Proceedings of the ACL conference workshop on computational approaches to Semitic languages.
- [9] Virga P, Khudanpur S. 2003. *Transliteration of Proper Names in Cross-lingual Information Retrieval*. Proceedings of ACL 2003 workshop on Multilingual and Mixed-language Named Entity Recognition. pp 57-64.
- [10] Jong-Hoon Oh, Key-Sun Choi. 2000. An English-Korean Transliteration Model Using Pronunciation and Contextual Rules. Proceedings of COLING.
- [11] Zhang Min, LI Haizhou SU Jian. 2004 *Direct Orthographical Mapping for Machine Transliteration*. Proceedings of the 20th international conference on Computational Linguistics
- [12] Gao Wei, Kam-Fai Wong, Wai Lam. 2004. *Phoneme-based transliteration of foreign names for OOV problem*. Proceedings of the First International Joint Conference on Natural Language Processing.
- [13] Kang B.J, Key-Sun Choi, 2000. *Automatic Transliteration and Back-transliteration by Decision Tree Learning*, Proceedings of the Second International Conference on Language Resources and Evaluation.
- [14] McCallum A, Freitag D, Pereira F. 2000. Maximum entropy Markov models for information extraction and segmentation. Proceedings of ICML.
- [15] Punyakanok V, Roth D. 2001. *The use of Classifiers in Sequential Inference*. Proceedings of the Conference on Advances in Neural Information Processing Systems.
- [16] Lafferty J, McCallum A, Pereira F. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. Proceedings of ICML. 282-289.
- [17] Taskar B, Lacoste-Julien S, Klein D. 2005. *A Discriminative Matching Approach to Word Alignment*. Proceedings of the Conference on Empirical Methods in Natural Language Processing.
- [18] Collins M. 2002. Discriminative Training for Hidden Markov Models: theory and Experiments with perceptron algorithms. Proceedings of EMNLP.