Improving transliteration accuracy using word-origin detection and lexicon lookup

Mitesh M. Khapra
IIT Bombay
miteshk@cse.iitb.ac.in

Pushpak Bhattacharyya
IIT Bombay
pb@cse.iitb.ac.in

Abstract

We propose a framework for transliteration which uses (i) a word-origin detection engine (*pre-processing*) (ii) a CRF based transliteration engine and (iii) a re-ranking model based on lexicon-lookup (*post-processing*). The results obtained for *English-Hindi* and *English-Kannada* transliteration show that the pre-processing and post-processing modules improve the top-1 accuracy by 7.1%.

1 Introduction

Machine transliteration is the method of automatically converting Out-Of-Vocabulary (OOV) words in one language to their phonetic equivalents in another language. An attempt is made to retain the original pronunciation of the source word to as great an extent as allowed by the orthographic and phonological rules of the target language. This is not a great challenge for language pairs like Hindi-Marathi which have very similar alphabetic and phonetic sets. However, the problem becomes non-trivial for language pairs like English-Hindi and English-Kannada which have reasonably different alphabet sets and sound systems.

Machine transliteration find its application in *Cross-Lingual Information Retrieval (CLIR)* and *Machine Translation (MT)*. In CLIR, machine transliteration can help in translating the OOV terms like proper names and technical terms which frequently appear in the source language queries (*e.g.* Jaipur in "Jaipur palace"). Similarly it can help improve the performance of MT by translating proper names and technical terms which are not present in the translation dictionary.

Current models for transliteration can be classified as *grapheme-based models*, *phoneme-based models* and *hybrid models*. Grapheme-based models like source channel model (Lee and Choi, 1998), Maximum Entropy Model (Goto et al., 2003), Conditional Random Fields (Veeravalli et al., 2008) and Decision Trees (Kang and Choi, 2000) treat transliteration as an orthographic process and try to map the source graphemes directly to the target graphemes. Phoneme based models like the ones based on Weighted Finite State Transducers (WFST) (Knight and Graehl, 1997) and extended Markov window (Jung et al., 2000) treat transliteration as a phonetic process rather than an orthographic process. Under this framework, transliteration is treated as a conversion from source grapheme to source phoneme followed by a conversion from source phoneme to target grapheme. Hybrid models either use a combination of a grapheme based model and a phoneme based model (Stalls and Knight, 1998) or capture the correspondence between source graphemes and source phonemes to produce target language graphemes (Oh and Choi, 2002).

Combining any of the above transliteration engines with pre-processing modules like word-origin detection (Oh and Choi, 2002) and/or post-processing modules like re-ranking using clues from monolingual resources (Al-Onaizan and Knight, 2002) can enhance the performance of the system. We propose such a framework which uses (i) language model based word-origin detection (ii) CRF based transliteration engine and (iii) a re-ranking model based on lexicon lookup on the target language (Hindi and Kannada in our case).

The roadmap of the paper is as follows. In section 2 we describe the 3 components of the proposed framework. In section 3 we present the results for English-Hindi and English-Kannada transliteration on the datasets (Kumaran and Kellner, 2007) released for NEWS 2009 Machine Transliteration Shared Task[1](Haizhou et al., 2009). Section 4 concludes the paper.

[1]https://translit.i2r.a-star.edu.sg/news2009/

Proceedings of the 2009 Named Entities Workshop, ACL-IJCNLP 2009, pages 84–87,
Suntec, Singapore, 7 August 2009.

2 Proposed framework for Transliteration

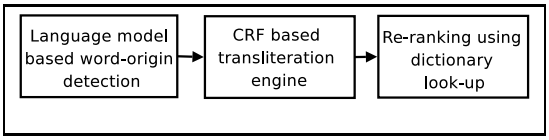

Figure 1: Proposed framework for transliteration.

2.1 Word Origin Detection

To emphasize the importance of Word Origin Detection we consider the example of letter *'d'*. When *'d'* appears in a name of Western origin (*e.g. Daniel, Durban*) and is not followed by the letter *'h'*, it invariably gets transliterated as Hindi letter ड, whereas, if it appears in a name of Indic origin (*e.g. Indore, Jharkhand*) then it is equally likely to be transliterated as द or ड. This shows that the decision is influenced by the origin of the word. The Indic dataset (Hindi, Kannada, and Tamil) for the Shared Task consisted of a mix of Indic and Western names. We therefore felt the need of training separate models for words of Indic origin and words of Western origin.

For this we needed to separate the words in the training data based on their origin. We first manually classified 3000 words from the training set into words of Indic origin and Western origin. These words were used as seed input for the bootstrapping algorithm described below:

1. Build two n-gram language models: one for the already classified names of Indic origin and another for the names of Western origin. Here, by n-gram we mean n-character obtained by splitting the words into a sequence of characters.

2. Split each of the remaining words into a sequence of characters and find the probability of this sequence using the two language models constructed in step 1.

3. If the probability of a word (i.e. a sequence of characters) is higher in the Indic language model than in the Western language model then classify it as Indic word else classify it as Western word.

4. Repeat steps 1-3 till all words have been classified.

Thus, we classified the entire training set into words of Indic origin and words of Western origin. The two language models (one for words of Indic origin and another for words of Western origin) thus obtained were then used to classify the test data using steps 2 and 3 of the above algorithm. Manual verification showed that this method was able to determine the origin of the words in the test data with an accuracy of 97%.

2.2 CRF based transliteration engine

Conditional Random Fields (Lafferty et al., 2001) are undirected graphical models used for labeling sequential data. Under this model, the conditional probability distribution of the target word given the source word is given by,

$$P(Y|X;\lambda) = \frac{1}{Z(X)} \cdot e^{\sum_{t=1}^{T}\sum_{k=1}^{K}\lambda_k f_k(Y_{t-1},Y_t,X,t)} \quad (1)$$

where,

$$\begin{aligned} X &= source\ word\ (English) \\ Y &= target\ word\ (Hindi, Kannada) \\ T &= length\ of\ source\ word\ (English) \\ K &= number\ of\ features \\ \lambda_k &= feature\ weight \\ Z(X) &= normalization\ constant \end{aligned}$$

CRF++[2] which is an open source implementation of CRF was used for training and decoding. GIZA++ (Och and Ney, 2000), which is a freely available implementation of the IBM alignment models (Brown et al., 1993) was used to get character level alignments for English-Hindi word pairs in the training data. Under this alignment, each character in the English word is aligned to zero or more characters in the corresponding Hindi word. The following features are then generated using this character-aligned data (here e_i and h_i are the characters at position i of the source word and target word respectively):

- h_i and e_j such that $i-2 \leq j \leq i+2$
- h_i and source character bigrams ($\{e_{i-1}, e_i\}$ or $\{e_i, e_{i+1}\}$)
- h_i and source character trigrams ($\{e_{i-2}, e_{i-1}, e_i\}$ or $\{e_{i-1}, e_i, e_{i+1}\}$ or $\{e_i, e_{i+1}, e_{i+2}\}$)

[2] http://crfpp.sourceforge.net/

- h_i, h_{i-1} and e_j such that $i - 2 \leq j \leq i + 2$
- h_i, h_{i-1} and source character bigrams
- h_i, h_{i-1} and source character trigrams

Two separate models were trained: one for the words of Indic origin and another for the words of Western origin. At the time of testing, the words were first classified as Indic origin words and Western origin words using the classifier described in section 2.1. The top-10 transliterations for each word were then generated using the correct CRF model depending on the origin of the word.

2.3 Re-ranking using lexicon lookup

Since the dataset for the Shared Task contains words of Indic origin there is a possibility that the correct transliteration of some of these words may be found in a Hindi lexicon. Such a lexicon containing 90677 unique words was constructed by extracting words from the Hindi Wordnet[3]. If a candidate transliteration generated by the CRF engine is found in this lexicon then its rank is increased and it is moved towards the top of the list. If multiple outputs are found in the lexicon then all such outputs are moved towards the top of the list and the relative ranking of these outputs remains the same as that assigned by the CRF engine. For example, if the 4th and 6th candidate generated by the CRF engine are found in the lexicon then these two candidates will be moved to positions 1 and 2 respectively. We admit that this way of moving candidates to the top of the list is adhoc. Ideally, if the lexicon also stored the frequency of each word then the candidates could be re-ranked using these frequencies. But unfortunately the lexicon does not store such frequency counts.

3 Results

The system was tested for English-Hindi and English-Kannada transliteration using the dataset (Kumaran and Kellner, 2007) released for NEWS 2009 Machine Transliteration Shared Task. We submitted one standard run and one non-standard run for the English-Hindi task and one standard run for the English-Kannada task. The re-ranking module was used only for the non-standard run as it uses resources (lexicon) other than those provided for the task. We did not have a lexicon for Kannada so were not able to apply the re-ranking module for English-Kannada task. The performance of the system was evaluated using 6 measures, *viz.*, Word Accuracy in Top-1 (ACC), Fuzziness in Top-1 (Mean F-score), Mean Reciprocal Rank (MRR), MAP_{ref}, MAP_{10} and MAP_{sys}. Please refer to the white paper of NEWS 2009 Machine Transliteration Shared Task (Haizhou et al., 2009) for more details of these measures.

Table 1 and Table 2 report the results[4] for English-Hindi and English-Kannada transliteration respectively. For English-Hindi we report 3 results: (i) without any pre-processing (word-origin detection) or post-processing (re-ranking) (ii) with pre-processing but no post-processing and (iii) with both pre-processing and post-processing. The results clearly show that the addition of these modules boosts the performance. The use of word-origin detection boosts the top-1 accuracy by around 0.9% and the use of lexicon lookup based re-ranking boosts the accuracy by another 6.2%. Thus, together these two modules give an increment of 7.1% in the accuracy. Corresponding improvements are also seen in the other 5 metrics.

4 Conclusion

We presented a framework for transliteration which uses (i) a word-origin detection engine (*pre-processing*) (ii) a CRF based transliteration engine and (iii) a re-ranking model based on lexicon-lookup (*post-processing*). The results show that this kind of pre-processing and post-processing helps to boost the performance of the transliteration engine. The re-ranking using lexicon lookup is slightly adhoc as ideally the re-ranking should take into account the frequency of the words in the lexicon. Since such frequency counts are not available it would be useful to find the web counts for each transliteration candidate using a search engine and use these web counts to re-rank the candidates.

[3] http://www.cfilt.iitb.ac.in/wordnet/webhwn

[4] Please note that the results reported in this paper are better than the results we submitted to the shared task. This improvement was due to the correction of an error in the template file given as input to CRF++.

Method	ACC	Mean F-score	MRR	MAP_{ref}	MAP_{10}	MAP_{sys}
CRF Engine *(no word origin detection, no re-ranking)*	0.408	0.878	0.534	0.403	0.188	0.188
CRF Engine + Word-Origin detection (no re-ranking) **Standard run**	0.417	0.877	0.546	0.409	0.192	0.192
CRF Engine + Word-Origin detection + Re-ranking **Non-Standard run**	**0.479**	**0.884**	**0.588**	**0.475**	**0.208**	**0.208**

Table 1: Results for English-Kannada transliteration.

Method	Accuracy (top1)	Mean F-score	MRR	MAP_{ref}	MAP_{10}	MAP_{sys}
CRF Engine + Word-Origin detection (no re-ranking) **Standard run**	0.335	0.859	0.453	0.327	0.154	0.154

Table 2: Results for English-Kannada transliteration.

References

B. J. Kang and K. S. Choi 2000. *Automatic transliteration and back-transliteration by decision tree learning*. Proceedings of the 2nd International Conference on Language Resources and Evaluation, 1135-1411.

Bonnie Glover Stalls and Kevin Knight 1998. *Translating Names and Technical Terms in Arabic Text*. Proceedings of COLING/ACL Workshop on Computational Approaches to Semitic Languages, 34-41.

Haizhou Li, A Kumaran, Min Zhang, Vladimir Pervouchine 2009. *Whitepaper of NEWS 2009 Machine Transliteration Shared Task*. Proceedings of ACL-IJCNLP 2009 Named Entities Workshop (NEWS 2009).

I. Goto and N. Kato and N. Uratani and T. Ehara 2003. *Transliteration considering context information based on the maximum entropy method*. Proceedings of MT-Summit IX, 125132.

J. S. Lee and K. S. Choi. 1998. *English to Korean statistical transliteration for information retrieval*. Computer Processing of Oriental Languages, 17-37.

John Lafferty, Andrew McCallum, Fernando Pereira 2001. *Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data*. In Proceedings of the Eighteenth International Conference on Machine Learning.

Jong-hoon Oh and Key-sun Choi 2002. *An English-Korean Transliteration Model Using Pronunciation and Contextual Rules*. Proceedings of the 19th International Conference on Computational Linguistics (COLING), 758-764.

Kevin Knight and Jonathan Graehl 1997. *Machine transliteration*. Computational Linguistics, 128-135.

Kumaran, A. and Kellner, Tobias 2007. *A generic framework for machine transliteration*. SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, 721-722.

Och Franz Josef and Hermann Ney 2000. *Improved Statistical Alignment Models*. Proc. of the 38th Annual Meeting of the Association for Computational Linguistics, pp. 440-447

P. F. Brown, S. A. Della Pietra, and R. L. Mercer 1993. *The mathematics of statistical machine translation: Parameter estimation*. Computational Linguistics, 19(2):263-311.

Sung Young Jung and SungLim Hong and Eunok Paek 2000. *An English to Korean transliteration model of extended Markov window*. Proceedings of the 18th conference on Computational linguistics, 383-389.

Suryaganesh Veeravalli and Sreeharsha Yella and Prasad Pingali and Vasudeva Varma 2008. *Statistical Transliteration for Cross Language Information Retrieval using HMM alignment model and CRF*. Proceedings of the 2nd workshop on Cross Lingual Information Access (CLIA) Addressing the Information Need of Multilingual Societies.

Yaser Al-Onaizan and Kevin Knight 2001. *Translating named entities using monolingual and bilingual resources*. ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, 400-408.