Transliteration of English Loanwords and Named-entities to Manipuri: Phoneme vs Grapheme representation

Lenin Laitonjam[†][§], Loitongbam Gyanendro Singh[§] and Sanasam Ranbir Singh[§]
[†]*Department of Computer Science and Engineering, National Institute of Technology Mizoram, Aizawl*
[§] *Department of Computer Science and Engineering Indian Institute of Technology Guwahati, Assam*
Email: {lenin.lai, gyanendrol9, ranbir}@iitg.ac.in

***Abstract*—Development of an effective system for transliterating loanwords and named-entities across orthographically very different languages is a challenging task. In general, such problems are addressed using two approaches; (i) dictionary-based mapping, and (ii) machine learning techniques. Most of the dictionary-based approaches focus on developing either phoneme or grapheme based mapping rules. In this paper, we investigate the effect of various transliteration models for transliterating English loanwords and named-entities to Manipuri language. First we compare performance between the state-of-the-art learning models namely RNN, LSTM and Bi-LSTM using seq2seq auto-encoder [1], and dictionary-based models. From various experimental observations, BiLSTM is found to be outperforming its counterparts. Further, we compare the performances of different transliteration methods over phoneme-based representation and grapheme-based representation. It is observed from experiments that grapheme-based representation outperforms its phoneme-based counterparts in both dictionary and learning based methods.**

Keywords-transliteration; grapheme; phoneme; manipuri;

I. INTRODUCTION

Natural embedding of loanwords from one language to another language has become a common phenomenon in today's writing system. Similar influence is also seen in naming people, locations and organizations (named-entities). If the morphological structure of the borrowed language is different from that of the primary language, automatic transliteration of the loanwords or named-entities to the target primary language is a non-trivial task. Considering the presence of high rate of loanwords in modern writing, development of an effective method for transliterating loanwords and named-entities is important for various NLP applications like machine translation (MT), cross-lingual information retrieval (CLIR), multi-lingual text processing, etc. This paper investigates the performance of various methods for transliterating English loanwords and named-entities to Manipuri language. Manipuri language (also known as *Meiteilon*)[1] is one of scheduled Indian languages and written using either *Bengali script* [2] or *Meitei mayek* [3]. This study considers Manipuri text written in Bengali script. To the best of our knowledge, this study is the first such attempt for transliterating English loanwords and named-entities to Manipuri.

Developing English to Manipuri transliteration system poses many challenges because of the distinct orthographic characteristics between the two languages. For instance, Bengali script uses 55 symbols to represent 38 phonemes in Manipuri [2], while English uses 26 symbols to represent 39 phonemes[4]. Further, Manipuri is syllabic in nature, while English is a phonetic language. There are two basic approaches for developing a transliteration system; namely (i) *dictionary-based models* [3][4] and (ii) *machine learning based models* [5][6][7]. These approaches use either phoneme or grapheme based representations. Phoneme-based representation approaches [8] require an intermediate phonetic representation, whereas grapheme-based approaches work directly on grapheme-to-grapheme mappings. Though grapheme based approaches are simpler in nature, it is effective when target languages are orthographically related[5][9]. In the studies [5][10], authors have exploited various machine learning models for developing transliteration system. In this paper, we systematically investigate the performance of dictionary-based approaches as well as machine learning based approaches by exploiting both the phoneme-based representation and grapheme-based representation. Since English and Manipuri are orthographically different languages, and morphological structure of Manipuri words are highly contextual dependent, dictionary-based approaches may not be suitable. For example, in transliteration pairs (কিলোমিটর, kilometer) and (সেপ্তেম্বর, september), English character *e* is mapped to different Bengali characters ি and ে respectively. In order to capture contextual dependency between character sequences, we transform the transliteration problem as a sequential labeling problem. In this study, we consider three neural network based sequential models namely RNN, LSTM and Bi-LSTM, and investigate their performance using seq2seq [1] auto-encoder. From the experimental results, we observe that grapheme-based representation outperforms its phoneme-based counterpart for both dictionary-based models and sequential models. Among the sequential models, Bi-LSTM outperforms both RNN and LSTM.

The rest of the paper is organized as follows: Section II presents related work. Section III gives the details about

[1]an official language used in Manipur (a north-eastern state of India).
[2]https://en.wikipedia.org/wiki/Bengali_alphabet
[3]https://en.wikipedia.org/wiki/Meitei_script
[4]https://en.wikipedia.org/wiki/CMU_Pronouncing_Dictionary

our dataset. In the section IV, we describe our proposed methods. Our experimental setup and results are discuss in Section V. Finally, section VI presents our conclusion and future work.

II. RELATED WORK

In literatures, transliteration tasks are generally performed either using phoneme/grapheme based dictionary maps or machine learning methods over large transliteration lexicon pairs. Dictionary based approach using phoneme maps can be found in the papers [11],[8] and [7]. For an effective phoneme based transliteration system, one needs to develop a large phoneme-phoneme pronunciation dictionary between the source and target languages which is an expensive exercise. The grapheme-grapheme based dictionary between source to target language is simpler to develop, but generally applied between orthographically similar languages. To take advantage of both the approaches, study [7] explore grapheme-to-phoneme model to apply over low resource settings. Instead of relying on expensive hand crafted rules, researchers have explored learning models in recent time [3][4]. Authors in [12][13] have considered discriminative statistical transliteration methods based on phrase-based statistical machine translation. Few successful works have also been reported using neural network architecture in [5][14][15]. Recently, authors in [10] have proposed an auto-encoder based machine transliteration model using Recurrent Neural Network and Convolutional sequence-to-sequence(Seq2Seq) architecture between English and Persian language pair over grapheme-based representation (NEWS 2018 Shared Task on Transliteration[5]). They claim to achieve better transliteration performance over other state-of-art methods. Motivated by this observation, our study adopts seq2seq framework and apply RNN, LSTM and Bi-LSTM to capture sequential dependency between the character sequence in their encoding and decoding stages.

To the best of our knowledge, this is the first effort for transliterating English loanwords and named-entities to Manipur. However, few studies on transliterating Manipuri text written in Bengali Script to Meitei Mayek have been reported in [16] and [17] using rule based methods. Considering that Manipuri text (same language) written in Bengali script and Meitei Mayek are orthogonally similar, obtaining such rules are simpler in nature. Whereas constructing rules between orthogonally dissimilar languages like English and Manipuri is an expensive task. In one of our earlier effort, building phoneme-based English to Manipuri pronunciation dictionary can be found in [18]. In this study, we develop rules for converting standard English CMU pronunciation dictionary[6] to correspond target Manipuri pronunciation using sequential labeling methods and use in developing Text-to-Speech Synthesis system. The phoneme representations used in this study are obtained using [18].

[5]http://workshop.colips.org/news2018/cfps.html
[6]https://en.wikipedia.org/wiki/CMU_Pronouncing_Dictionary

Table I. DATASET DESCRIPTION

TOTAL SENTENCE PAIRS	9892
Average English sentence length	25.65
Average Manipuri sentence length	25.06
Unique Transliteration Lexicon Pairs	6035
Named-entities	3402
Loanwords	2633

Table II. TRANSLITERATION PAIR SAMPLES

English Word	**Manipuri Transliteration**
chitra	চিত্রা
warden	ৱার্ডেন
tropic	ট্রোপিক্

III. DATASET

This paper considers a English-to-Manipuri parallel corpus publicly available at http://tdil-dc.in. This dataset is originally distributed for building Manipuri machine translation system in Tourism domain. To create dataset for our study, we extracted English loanwords and named-entities from this parallel corpus. Detailed description of the dataset is shown in Table I. It consist of 9892 parallel sentences and 6035 unique transliterated English loanwords and named-entities. Out of the 6035 lexicon pairs, 3402 are named-entities and remaining 2633 are loanwords. A sample transliteration lexicon pairs is shown in Table II.

IV. PROPOSED METHODS

In this paper, we investigate performances of both dictionary-based and machine learning based methods for transliterating English loanwords and named-entities to Manipuri language in Bengali script. For all these methods, we have considered both phoneme as well as grapheme-based representations. In this section, we first discuss the proposed dictionary-based methods (for both phoneme and grapheme) and then different classification frameworks used for this study.

A. Dictionary-based approaches

1) Phoneme-based approach: Given a language pair (English and Manipuri in our case), phoneme-based approaches [11], [8] try to capture phonetic relationship between the source and target language pairs. The phonetic relationships are then used to establish orthographic association between the language pair. Phoneme based representation generally involves two steps. First, it converts the source grapheme to an intermediate phonetic representation. Second, it again maps the intermediate phonetic representation to the grapheme of the target language. Figure 1 shows the proposed method for transliterating English words to Manipuri using phoneme-based rules. Given an English word (or named-entity), it is first transformed the word into a sequence of phonemes of the target language using a pre-defined dictionary. The intermediate phoneme-sequence is then mapped to corresponding grapheme sequence of the target language

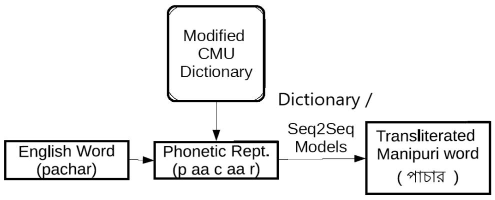
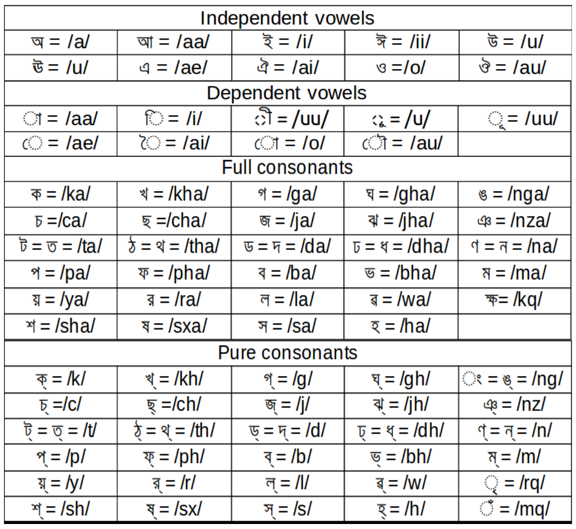


Figure 1. Phoneme-based Approach



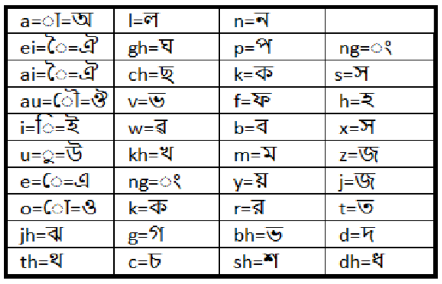
Independent vowels				
অ = /a/	আ = /aa/	ই = /i/	ঈ = /ii/	উ = /u/
ঊ = /u/	এ = /ae/	ঐ = /ai/	ও =/o/	ঔ = /au/
Dependent vowels				
◌া = /aa/	ি◌ = /i/	◌ী = /uu/	◌ু = /u/	◌ূ = /uu/
ে◌ = /ae/	ৈ◌ = /ai/	ে◌া = /o/	ে◌ৗ = /au/	
Full consonants				
ক = /ka/	খ = /kha/	গ = /ga/	ঘ = /gha/	ঙ = /nga/
চ =/ca/	ছ =/cha/	জ = /ja/	ঝ = /jha/	ঞ = /nza/
ট = ত = /ta/	ঠ = থ = /tha/	ড = দ = /da/	ঢ = ধ = /dha/	ণ = ন = /na/
প = /pa/	ফ = /pha/	ব = /ba/	ভ = /bha/	ম = /ma/
য় = /ya/	র = /ra/	ল = /la/	ৱ = /wa/	ক্ষ= /kq/
শ = /sha/	ষ = /sxa/	স = /sa/	হ = /ha/	
Pure consonants				
ক্ = /k/	খ্ = /kh/	গ্ = /g/	ঘ্ = /gh/	◌ং = ঙ্ = /ng/
চ্ =/c/	ছ্ =/ch/	জ্ = /j/	ঝ্ = /jh/	ঞ্ = /nz/
ট্ = ত্ = /t/	ঠ্ = থ্ = /th/	ড্ = দ্ = /d/	ঢ্ = ধ্ = /dh/	ণ্ = ন্ = /n/
প্ = /p/	ফ্ = /ph/	ব্ = /b/	ভ্ = /bh/	ম্ = /m/
য়্ = /y/	র্ = /r/	ল্ = /l/	ৱ্ = /w/	◌ৃ = /rq/
শ্ = /sh/	ষ্ = /sx/	স্ = /s/	হ্ = /h/	◌ঁ = /mq/

Figure 2. Manipuri phoneme mapping table

to obtain the target transliteration. In this approach, generating cross lingual phoneme dictionary is an expensive operation. In our earlier paper [18], a method for adapting CMU pronunciation dictionary from English to Manipuri is proposed using sequence labeling methods such as CRF and obtained a F-score measure of 0.991 for phone level classification and 0.93 word level accuracy. Considering the high accuracy reported in this paper, we consider the modified CMU dictionary proposed in [18] for generating English to Manipuri phoneme-based mapping.

Further in another study [19], we propose phoneme-to-grapheme maps for Manipuri language in Bengali scripts. This mapping is shown in figure 2. For generating the target grapheme from the intermediate phoneme representation, we use the same mapping table as described in figure 2. Since a sequence of a vowel following a consonant form a cluster element in Manipuri writing (true for most of the Indian languages), we further apply the following rules while converting from the phoneme to grapheme through figure 2. We map a vowel phoneme to a dependent vowel grapheme when it follows a consonant phoneme. Similarly, when a vowel phoneme `/a/` follows a consonant phoneme, the consonant phoneme is mapped to a full consonant grapheme.

2) Grapheme-based approach: Grapheme-based approaches [5], [10] try to obtain bilingual orthographic associations between the language pair, and directly maps a source grapheme to a target grapheme without intermediate phoneme transformation. Figure 3 shows the



a=◌া=অ	l=ল	n=ন	
ei=ৈ◌=ঐ	gh=ঘ	p=প	ng=◌ং
ai=ৈ◌=ঐ	ch=ছ	k=ক	s=স
au=ে◌ৗ=ঔ	v=ভ	f=ফ	h=হ
i=ি◌=ই	w=ৱ	b=ব	x=স
u=◌ু=উ	kh=খ	m=ম	z=জ
e=ে◌=এ	ng=◌ং	y=য়	j=জ
o=ে◌া=ও	k=ক	r=র	t=ত
jh=ঝ	g=গ	bh=ভ	d=দ
th=থ	c=চ	sh=শ	dh=ধ

Figure 3. Manipuri grapheme mapping table

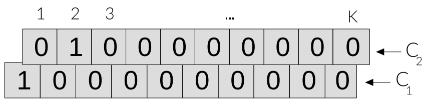


Figure 4. Character embedding for training Seq2Seq model

grapheme-grapheme mapping table between English to Manipuri. Given an English word, corresponding Manipuri transliteration is generated using this table. For example, a named-entity word *Lenin* is mapped as লেনিন using the considered mapping table in our grapheme-based approach. As discuss in phoneme based transformation above, a vowel is converted to dependent vowel when it follows after a consonant.

B. Machine learning methods

With the advancement of various learning methodologies, researchers have successfully used machine learning models for developing transliteration system (instead of relying on expensive hand-crafted rules). Given a collection of English to Manipuri lexicon pairs, machine learning methods tend to learn association between the two languages. Given a sequence of characters (in the source language), the task of transliteration is to generate a sequence of character in the target language. Considering the nature of transformation, we can consider the task of automatic transliteration as a sequence labeling problem [5][14][15][20].

Recently, authors in [1] propose a sequence-to-sequence (Seq2Seq) based Neural Machine Translation (NMT) framework for converting an English sentence to corresponding sentence in French. Given an input sequence of words (source sentence in English), it learns sequential characteristics of the target word sequence (target sentence in French). A Seq2Seq model is an auto-encoder primarily consists of two parts i.e. an *encoder* for learning an intermediate representation and a *decoder* for generating target sequence from the intermediate representation. In this study, we have used Recurrent Neural Network (RNN), Long Short Term Memory (LSTM) and Bi-directional LSTM (BiLSTM) neural network models for setting the encoder and decoder. We use Keras Deep learning python libraries[7] for implementing the Seq2Seq models.

[7]https://keras.io

To train the proposed Seq2Seq model, we have represented the training set into 3 types of multidimensional arrays, namely *encoder_input_data, decoder_input_data* and *decoder_target_data* to represent character sequences for every word pairs <input, target> in the training set. Each array is composed of the following dimensions: (*total training samples, current input word length, total number of characters*) for each language. We consider the phonemes of the input words and the graphemes of the target words as the characters for the phoneme-based approach. While for the grapheme-based approach, we consider the graphemes of each words in the training sets as the characters. The first array i.e. *encoder_input_data* stores the character embedding vector for each input samples in the training set. Similarly the second array i.e. *decoder_input_data* stores the character embedding vector for each target samples in the training set. These two arrays are the input arrays for representing the whole word pairs individually. The third array i.e. *decoder_target_data* is same as the *decoder_input_data* with an offset of one timestep. We can access the ith word from input arrays as (i,t,C_j) where t is the tth character c_j in *word*(i) and C_j is the one-hot vector representation of c_j character. Figure 4 shows the one-hot vector embedding of character c_2 for *word*(i) i.e. (i,2,C_2) of the input arrays. K is the total number of unique characters present in each types of training words. The difference of *decoder_input_data* and *decoder_target_data* arrays is that for each (i,t,C_j) in *decoder_input_data*, it is (i,t+1,C_j) in *decoder_target_data* array.

We feed *encoder_input_data* and *decoder_input_data* to the encoder and decoder DNNs respectively. The internal states for both encoder and decoder have same dimension in each DNNs. The encoder processes its input sequences and store the weights of its internal states. These weights serve as the initial context for the decoder to process its input sequences. The decoder then try to predict the candidate character sequences through the provided context information of the encoder. *decoder_target_data* acts as the target data for training the DNNs of the Seq2Seq model.

V. EXPERIMENTAL SETUP AND RESULTS

For all the experimental discussion reported in this section, we consider the dataset described in Table I. Out of 6035 unique lexicon pairs, we randomly select 90% as the training and 10% as the testing sets. While writing Manipuri using Bengali script, some of the vowels and consonants are used interchangeably. Table III shows the set of interchangeable vowels and consonants. Considering such characteristics, we further normalize the Manipuri text by resolving the interchangeable characters i.e., mapping all the interchangeable character set to a single representative character.

A. Results and Discussion

This section discusses the experimental results of various models proposed in this paper. We have considered Character Error Rate (CER) and Word Accuracy as our evaluation matrices. Table IV and V shows CER and Word Accuracy for our models respectively. Based on various experimental evaluations, we observe the following important observations:

Table III. INTERCHANGEABLE CHARACTERS

ী : ি	ণ : ন	ঠ : থ	স : ষ : শ	
ূ : ু	ঢ : ধ	ড : দ	ঢ় : র	ট : ত

Table IV. CER (%) FOR VARIOUS TRANSLITERATION MODELS

	Dictionary	**RNN**	**LSTM**	**BiLSTM**
Phoneme-based	40.60	83.40	49.13	33.11
Normalized Phoneme-based	33.23 (18.15% ↓)	83.40 (0% ↓)	47.19 (3.95% ↓)	30.39 (8.21% ↓)
Grapheme-based	34.70	83.20	42.87	27.03
Normalized Grapheme-based	29.72 (14.35% ↓)	82.88 (0.3% ↓)	40.29 (6.02% ↓)	24.17 (10.58% ↓)

- Grapheme-based models outperform their phoneme-based counterparts.
- Among the three Seq2Seq models, BiLSTM outperforms its counterparts for both phoneme-based as well as grapheme-based approaches.
- The proposed dictionary-based models outperform Seq2Seq models using RNN and LSTM.
- Text normalization helps in improving the transliteration performance. It achieves an improvement upto 72.64% over phoneme representation and upto 25.93% over grapheme representation in word accuracy. Similar observation is true for CER.

In table IV, we observe that grapheme-based models outperform phoneme-based models by almost 18% on average. Among the three Seq2Seq models, BiLSTM over the normalized dataset outperforms its counterparts. We also observe that BiLSTM outperforms the dictionary-based methods over the normalized text by almost 23% in CER.

We further investigate the transliterated text obtain from BiLSTM over the normalized text. We observe that majority of the errors are due to single character error, constituting about 31.48% of the total word errors. An interesting observation is that majority of the single character errors occur at the last position (30%). These are mainly due to ambiguous nature of pronouncing last character *s/es* in English plural words to Manipuri. In Manipuri language, plural are implied by adding suffixes to its singular word as a morphological inflection or by using another word representing plural, such as *m*any, beside the singular word. We are able to reduce the CER from 24.17% to 23.20% after ignoring the single character error for all the plural words, i.e. assuming the transliterated words to be correct.

Another interesting observation is that the models over the normalized text is able to capture various orthographic relationships, even producing different spelled word with equivalent pronunciation. Example of such words are shown in Table VI. Assuming such words as correct, word

Table V. WORD ACCURACY (%) FOR VARIOUS TRANSLITERATION MODELS

	Dictionary	RNN	LSTM	BiLSTM
Phoneme-based	7.64	0	4.73	14.56
Normalized Phoneme-based	13.19 (72.64% ↑)	0	5.26 (11.20% ↑)	16.99 (16.69% ↑)
Grapheme-based	12.53	0	9.68	22.10
Normalized Grapheme-based	14.36 (14.60% ↑)	0	12.19 (25.93% ↑)	27.54 (24.62% ↑)

Table VI. DIFFERENT SPELLED WORD WITH EQUIVALENT PRONUNCIATION

English Word	**Correct Spelling**	**Equivalent pronunciation**
aiyapa	আইয়াপা	অয়য়াপা
ghagra	ঘাগ্রা	ঘাগররা
bluegrass	ব্লুগ্রাশ	ব্লুগররাশ

accuracy of dictionary-based and BiLSTM model over normalized text is shown in Table VII.

Though, our grapheme-based models perform comparatively better, it experiences certain limitations. In some instances, it is unable to capture some of the basic Manipuri linguistic features like two dependent vowels cannot be placed one after another and a dependent vowel cannot exist independently. Such error can be rectified if we are able to incorporate the linguistic features to the Seq2Seq models. We open this problem for future study.

VI. CONCLUSION AND FUTURE WORK

We have explored various approaches for obtaining an effective transliteration model. We show that relatively cheaper grapheme-based models perform comparatively better than phoneme-based ones even for orthographically distinct language pair. A primary reason could be that defining an appropriate intermediate phonetic representation that captures orthographic relationship is much more challenging than finding it directly from grapheme-to-grapheme associations. Our best model gives an accuracy of only 27.54% with CER of 24.17% which shows that there is still lots of room for improvement. In future, we would like to try other models and also add language specific contextual rules to improve accuracy as well as CER.

Table VII. WORD ACCURACY (%) AFTER CONSIDERING EQUIVALENT PRONUNCIATION

	Dictionary	BiLSTM
Normalized Phoneme-based	16.32 (23.73% ↑)	21.05 (23.90% ↑)
Normalized Grapheme-based	23.59 (64.28% ↑)	34.56 (25.49% ↑)

REFERENCES

[1] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[2] L. S. Singh, K. Thaoroijam, and P. K. Das, "Written manipuri (meiteiron) from phoneme to grapheme." *Language in India*, vol. 7, no. 6, 2007.

[3] J.-H. Oh and K.-S. Choi, "An english-korean transliteration model using pronunciation and contextual rules," in *Proceedings of the 19th international conference on Computational linguistics-Volume 1*. Association for Computational Linguistics, 2002, pp. 1–7.

[4] D. Bhalla, N. Joshi, and I. Mathur, "Rule based transliteration scheme for english to punjabi," *arXiv preprint arXiv:1307.4300*, 2013.

[5] A. Kunchukuttan, M. Khapra, G. Singh, and P. Bhattacharyya, "Leveraging orthographic similarity for multilingual neural transliteration," *Transactions of the Association for Computational Linguistics*, vol. 6, pp. 303–316, 2018.

[6] Y. Jia, D. Zhu, and S. Yu, "A noisy channel model for grapheme-based machine transliteration," in *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration*. Association for Computational Linguistics, 2009, pp. 88–91.

[7] N. T. Le and F. Sadat, "Low-resource machine transliteration using recurrent neural networks of asian languages," in *Proceedings of the Seventh Named Entities Workshop*, 2018, pp. 95–100.

[8] P. Virga and S. Khudanpur, "Transliteration of proper names in cross-lingual information retrieval," in *Proceedings of the ACL 2003 workshop on Multilingual and mixed-language named entity recognition-Volume 15*. Association for Computational Linguistics, 2003, pp. 57–64.

[9] H. Sajjad, N. Durrani, H. Schmid, and A. Fraser, "Comparing two techniques for learning transliteration models using a parallel corpus," in *Proceedings of 5th International Joint Conference on Natural Language Processing*, 2011, pp. 129–137.

[10] S. Kundu, S. Paul, and S. Pal, "A deep learning based approach to transliteration," in *Proceedings of the Seventh Named Entities Workshop*, 2018, pp. 79–83.

[11] J.-S. Kuo, H. Li, and Y.-K. Yang, "Learning transliteration lexicons from the web," in *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2006, pp. 1129–1136.

[12] A. Finch and E. Sumita, "A bayesian model of bilingual segmentation for transliteration," in *International Workshop on Spoken Language Translation (IWSLT) 2010*, 2010.

[13] G. Nicolai, B. Hauer, M. Salameh, A. St Arnaud, Y. Xu, L. Yao, and G. Kondrak, "Multiple system combination for transliteration," in *Proceedings of the Fifth Named Entity Workshop*, 2015, pp. 72–77.

[14] R. Grundkiewicz and K. Heafield, "Neural machine translation techniques for named entity transliteration," in *Proceedings of the Seventh Named Entities Workshop*, 2018, pp. 89–94.

[15] A. Finch, L. Liu, X. Wang, and E. Sumita, "Neural network transduction models in transliteration generation," in *Proceedings of the Fifth Named Entity Workshop*, 2015, pp. 61–66.

[16] K. Nongmeikapam, N. H. Singh, S. Thoudam, and S. Bandyopadhyay, "Manipuri transliteration from bengali script to meitei mayek: A rule based approach," in *Information Systems for Indian Languages*. Springer, 2011, pp.

195–198.

[17] T. D. Singh, "Bidirectional bengali script and meetei mayek transliteration of web based manipuri news corpus," in *the Proceedings of the 3rd Workshop on South and Southeast Asian Natural Language Processing (SANLP) of COLING*, 2012, pp. 181–189.

[18] R. Saikia and S. R. Singh, "Generating manipuri english pronunciation dictionary using sequence labelling problem," in *Asian Language Processing (IALP), 2016 International Conference on*. IEEE, 2016, pp. 67–70.

[19] L. G. Singh, L. Laitonjam, and S. R. Singh, "Automatic syllabification for manipuri language," in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, 2016, pp. 349–357.

[20] A. Finch, L. Liu, X. Wang, and E. Sumita, "Target-bidirectional neural models for machine transliteration," in *Proceedings of the Sixth Named Entity Workshop*, 2016, pp. 78–82.