Leveraging Alignment and Phonology for low-resource Indic to English Neural Machine Transliteration

Parth Patel[1], Manthan Mehta[2], Pushpak Bhattacharyya[1] and Arjun Atreya[1]
{parthpatel, pb, arjun}@cse.iitb.ac.in, f20170408@pilani.bits-pilani.ac.in
[1]Department of Computer Science & Engineering
Indian Institute of Technology Bombay, Mumbai, India
[2] Department of Computer Science & Information Systems
Birla Institute of Technology & Science, Pilani, India

Abstract

In this paper we present a novel transliteration technique based on Orthographic Syllable *(OS)* segmentation for low-resource Indian languages *(ILs)*. Given that alignment has produced promising results in Statistical Machine Transliteration systems and phonology plays an important role in transliteration, we introduce a new model which uses alignment representation similar to that of IBM model 3 to pre-process the tokenized input sequence and then use pre-trained source and target OS-embeddings for training. We apply our model for transliteration from *ILs* to English and report our accuracy based on Top-1 Exact Match. We also compare our accuracy with a previously proposed Phrase-Based model and report improvements.

1 Introduction

The process of transliteration is defined in Zhang et al. (2012) as *"the conversion of a given name in the source language (a text string in source script) to a name in the target language (another text string in target script), such that the target language name is: (i) phonemically equivalent to the source name, (ii) conforms to the phonology of the target language, and (iii) matches the user intuition of the equivalent of the source language name in the target language, considering the culture and orthographic character usage in the target language"*. This definition of transliteration is apt in the context of Machine Translation since it employs transliteration as a subsystem to handle Named Entities (NEs). We are interested in solving the transliteration problem for Indian to English language pairs and in this paper, we demonstrate the use of OS and pre-trained embeddings to overcome the data sparsity problem that arises in low-resource languages.
The structure of the paper is as follows: Section 2 presents the state of the art on machine transliteration. In section 3 and 4, we describe some background and our proposed approach. Then, in section 5, we present our experiments and results. Finally, in section 6, we present our conclusions and in section 7, we express gratitude to our supporters.

2 Related Work

Arbabi et al. (1994) proposed the very first transliteration system for Arabic to English transliteration. In 1998, Knight and Graehl. (1998) proposed a statistical based approach that back transliterates English to Japanese Katakana which was later adopted for Arabic to English back transliteration by Stalls and Knight. (1998). In 2000, three independent research teams proposed English-to-Korean transliteration models. Other series of work on transliteration has focused on character as a unit of transliteration and using Recurrent Neural Networks. Neural network-based system in the 2016 was proposed by Finch et al. (2016) for multiple language pairs. They used Bi-directional LSTMs for good prefix and suffix generation and were able to surpass the state-of-the-art results of previous systems on the same datasets. Kunchukuttan et al. (2018); Le et al. (2019) used standard encoder-decoder architecture (with attention mechanism (Bahdanau et al., 2014)). Until recently, the best-performing solutions were discriminate statistical transliteration methods based on OS-based statistical machine transliteration (Atreya, 2016) for Indian to English language pairs. We focus on applying OS as a transliteration unit on encoder-decoder architecture (with attention) (Luong et al., 2015).

3 Background Knowledge

We first describe the Orthographic syllables, which form the essence of this transliteration module, introducing a new technique for word segmentation, following which we will formulate the probabilistic model for Grapheme-to-Grapheme alignment, explaining the method to position each orthographic syllable in the word.

3.1 Orthographic Syllables

Indic languages possess greater grapheme to phoneme consistency as compared to English(Atreya, 2016). However, the syllable boundary identification for segmentation of an Indic language word into a list of syllables is extremely challenging because of the presence of Schwa (short 'a' vowel preceded by a consonant unless specified otherwise) and diphthongs (sound formed by combination of two vowels) in the syllable unit. In this work, we have used a variant of Syllable as a unit, called Orthographic Syllable which essentially is 'Syllable $-$(*minus*) Coda' (See Figure:1). The algorithm used for the OS segmentation is presented in Algorithm 1.

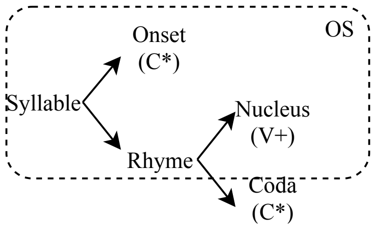

Figure 1: Structure of Orthographic Syllable where the left over Coda concatenates with the Onset of the next OS

3.2 Grapheme-to-Grapheme Alignment

Le et al. (2019) and Yao and Zweig (2015) have reported that, in grapheme-to-phoneme alignments, the grapheme possesses the fertility property similar to IBM Model 3 i.e. it can map to either a null or single or compound phonemes. We assume that same holds true for grapheme-to-grapheme alignments. Since, the aim of alignment is to find a grapheme sequence Y defined by $Y = \{p_1, p_2, ..., p_N\}$, given an OS segmented grapheme sequence X defined by $X = \{o_1, o_2, ..., o_M\}$. Essentially, the problem can be seen as finding the optimal grapheme sequence $\hat{Y}$, which maximizes its conditional probability, as in Equation 1. Since $p(X)$ is independent of the grapheme sequence Y, we can simplify the equation 2 to get Equation 3.

$$\hat{Y} = \arg\max_Y p(Y|X) \quad (1)$$

$$\hat{Y} = \arg\max_Y \frac{(X|Y)p(Y)}{p(X)} \quad (2)$$

$$\hat{Y} = \arg\max_Y p(X|Y)p(Y) \quad (3)$$

Mathematically, given X, Y, and an alignment A, the posterior probability $p(P|O, A)$ is estimated as follows:

$$p(Y|X, A) \approx \prod_{n=1}^{N} p(p_n|p_{n-1}^{n-k}, o_{n-k}^{n+k}) \quad (4)$$

where k is the context window size and n is the alignment position index.
We use expectation-maximization as described in Jiampojamarn et al. (2007) for re-aligning the input sequence after OS segmentation.

Algorithm 1 Orthographic Syllable segmentation . Consonant and Vowel are represented by C and V respectively

```
1: procedure SEGMENT(word)    ▷ Split word based on regular expression: $C^*V^+$
2:     $seg\_os \leftarrow$ “”
3:     $prev\_vowel \leftarrow False$
4:     for each character $c$ in $word$ do
5:         if $prev\_vowel$ and ($c$!=$vowel$) then
6:             $prev\_vowel \leftarrow False$
7:             $seg\_os \leftarrow seg\_os$ + “#”
8:         end if
9:         if $c = vowel$ then
10:            $prev\_vowel \leftarrow True$
11:        end if
12:        $seg\_os \leftarrow seg\_os$ + $c$
13:    end for
14:    return $seg\_os.split$(“#”)
15: end procedure
```

4 Approach

In this section, we present our approach, followed by a description of our experimental setup, describing the data gathering and cleaning, followed by the model configurations and then the evaluation technique.

4.1 Proposed Approach

Our approach for Indic to English Neural Machine Transliteration consists of 4 steps: *(1) orthographic syllable segmentation, (2) modification of OS-segmented input sequences based on alignment representation, (3) creation of orthographic syllable embeddings with aligned input sequences as input and (4) then we train an RNN-based machine transliteration model.* The whole process is illustrated in Figure 2.

4.2 Experimental Data

We run our experiments on baby names dataset available in multiple *IL* and English from India Child Names website[1] and Bachpan website[2]. The bilingual dataset for learning is divided into training, development, and test-sets at a ratio of 90%, 5% and 5% respectively. The details about the dataset are mentioned in Table 1.

4.3 Model Configuration

We use the *m-2-m aligner*[3] toolkit (Jiampojamarn et al., 2007) to align the training data at OS level. We choose $m = 2$ similar to (Le et al., 2019) for alignment. For the pre-trained source and target OS embeddings, we apply gensims[4] toolkit (Řehůřek and Sojka, 2010) with dimension size of 100, 200, and 300, a window size of 3, and the skip-gram option.

For model training, we apply OpenNMT-py[5] toolkit (Klein et al., 2017) to train our transliteration model. In the transliteration system configuration, we run our model with *Adam* optimizer and use Luong et al. (2015) attention with two learning rates 0.01 and 0.001 for 50000 training steps. We also use 2, 3, and 4 layered encoder-decoder networks(LSTMs) each with vector sizes of 100, 200, and 300 and report the top accuracy values for multiple language pairs.

[1] `www.indiachildnames.com`

[2] `www.bachpan.com`

[3] `https://github.com/letter-to-phoneme/m2m-aligner`

[4] `https://radimrehurek.com/gensim/models/fasttext.html`

[5] `https://github.com/OpenNMT/OpenNMT-py`

xx-en pair	train	dev	test	total
Assamese (as)	95308	5295	5295	105897
Bengali (bn)	71832	3991	3991	79813
Gujarati (gu)	16599	923	923	18443
Hindi (hi)	43074	2393	2393	47860
Kannada (kn)	16601	923	923	18445
Malayalam (ml)	15721	874	874	17467
Marathi (mr)	46908	2606	2606	52120
Punjabi (pa)	44737	2486	2486	49707
Tamil (ta)	16393	911	911	18214
Telugu (te)	19970	1110	1110	22188

Table 1: Dataset details for 10 *IL* where xx represents *IL* code mentioned in parenthesis of column 1

4.4 Evaluation Technique

We use Top-1 Exact Match accuracy as the evaluation metric (Banchs et al., 2015). This is one of the metrics used in the NEWS shared tasks on transliteration.

5 Results

We discuss and analyse the results of our experiments, indicating the major improvements and scope of improvement for our approach.

5.1 Results on Test Data

To evaluate our proposed approach, we have implemented three systems (Table 2):

1. Baseline System: We reproduce the results of Atreya (2016) using MOSES[6] toolkit (Koehn et al., 2007) for our experiments along with GIZA++[7] (Och and Ney, 2003) for learning alignments.

2. System 1: Encoder-Decoder LSTM(Klein et al., 2017) + Attention Mechanism(Luong et al., 2015) + OS segmentation(Atreya, 2016)+ one hot encoding as the encoding mechanism.

3. System 2: Encoder-Decoder LSTM(hidden sizes of 128 and 256 were tried) + Attention Mechanism + OS segmentation + pre-trained source and target OS embeddings(sizes of 100, 200 and 300 were used as embedding

[6] `https://github.com/moses-smt/mosesdecoder`

[7] `https://github.com/moses-smt/mgiza`

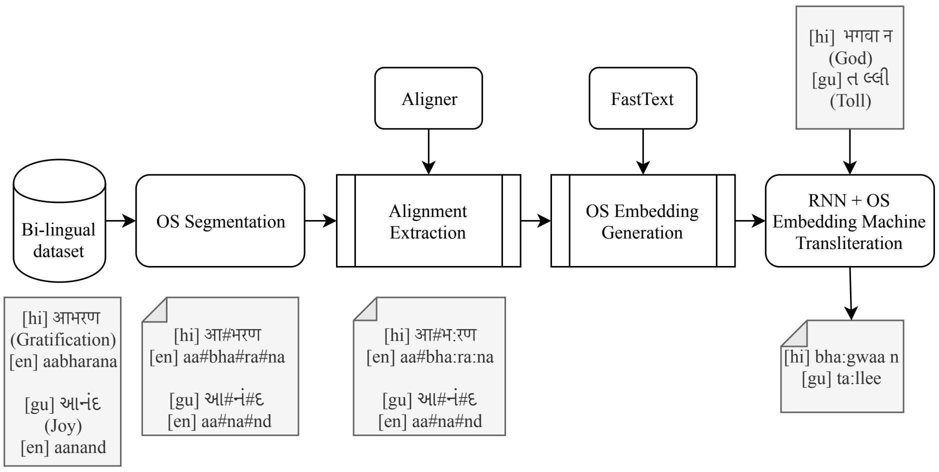

Figure 2: System Architecture for Indic to English Neural Machine Transliteration

Exp.	DS	Indic (xx) to English (en) Language pair									
		as	bn	gu	hi	kn	ml	mr	pa	ta	te
Baseline		50.68	37.75	48.87	57.21	48.94	48.49	45.77	39.28	35.39	46.99
System 1	128	81.92	75.8	73.58	**63.52**	72.7	71.82	67.98	66.51	65.38	69.19
	256	80.43	75.56	73.75	62.96	73.96	71.19	67.43	66.45	65.76	70.22
System 2	128	**82.97**	**78.42**	72.59	62.68	74.29	**74.65**	68.83	**68.73**	65.7	**71.59**
	256	81.54	77.21	**73.84**	62.73	**75.48**	74.54	**69.2**	68.16	**66.77**	69.78

Table 2: Top-1 accuracy figures of xx-en language pairs with 128 and 256 dimension size(DS)

sizes). The learning rate used here was 0.001 with Adam optimizer as already mentioned in section 4.3.

As evident from Table 2, our systems 1 and 2 increase the Top-1 accuracy by at-least **50%**. We list the following observations:

- All language pairs perform better with a learning rate of 0.001 and a 2-layered LSTM.
- We claim that the dimension size is inversely proportional to the size of the dataset for Indic languages. This is supported by the fact that {gu, kn, ta}-en language pairs have smaller dataset size and perform better for LSTM with dimension size of 256. On the other hand, the {as, bn, hi, ml, pa, te}-en language pairs have a larger dataset and perform better with dimension size of 128.

5.2 Error Analysis

Table 3 shows top-10 prediction errors along with actual and predicted output examples.

	y	$\hat{y}$	Count	Expected Word	Output Word
1	ee	i	832	ha mee d	haa mi d
2	aa	a	667	ko maa n	ko ma n
3	i	ee	567	haa mi d	ha mee d
4	th	t	288	vi dva thi	vi dva ti
5	w	v	202	i swa r	i sva r
6	t	th	187	ra nti ka	ra nthi ka
7	a	aa	174	ha mee d	haa mi d
8	v	w	158	vo to n	wo to n
9	c	k	107	mou ni ca	mou ni ka
10	k	c	55	ana mi ka	ana mi ca

Table 3: Top-10 Most confused vowels across all language pairs. y represents the expected output whereas $\hat{y}$ represents the actual predicted output

The most frequent error the system makes is confusing long ई(**E**) sound with a short इ(**e**) and have only predicted correctly 487 times. The characters थ(**th**) and त(**t**), both unaspirated and aspirated consonants, are also mistakenly substituted. *Schwa* present at the end of an OS also presents a challenge for the prediction since *IL* words are almost always suffixed by a short अ(**a**) sound(unless otherwise specified explicitly by using ◌्) that is non-existent in English words. This is also language dependent since राज(raj, rule) from Hindi to English should be transliterated as *Raj* whereas from Dravidian(ta, te, kn, ml) languages should be *Raja*. Similarly, even words of the same language can have two different predictions like मा*(mother)* have *ma* and *maa* which are both correct with respect to English phonology. The characters *w* and *v* are the sounds that both maps to the same akshar of Indo-Aryan languages and are often very difficult to differentiate.

6 Conclusion and Future Work

We show that using pre-trained OS-embeddings on neural encoder-decoder architecture involving OS tokenization outperforms the baseline system by a significant margin. The results also support our claim that phonology and alignment play an important role in increasing the accuracy of transliteration. The reason for the improvement could be learning the Akshar (a combination of vowel and consonant) representation by encoder network and the ability to learn canonical spellings in English.

Given the benefits of using alignment and OS embeddings for low-resource *ILs*, we intend to explore *IL* to *IL* transliteration with and without English as a pivot.

7 Acknowledgement

We would like to show our gratitude to Ministry of Electronics And IT (MEITY[8]) and our colleagues from Center for Indian Language Technology (CFILT[9]) who provided insight, expertise, and resources that greatly assisted the research, and we thank 2 "anonymous" reviewers for comments that improved the manuscript.

References

Mansur Arbabi, Scott M Fischthal, Vincent C Cheng, and Elizabeth Bart. 1994. Algorithms for arabic name transliteration. *IBM Journal of research and Development*, 38(2):183–194.

Arjun Atreya. 2016. Structure cognizant multilingual query expansion in resource scarce languages. Ph.d thesis, IIT Bombay, April.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Rafael E Banchs, Min Zhang, Xiangyu Duan, Haizhou Li, and A Kumaran. 2015. Report of news 2015 machine transliteration shared task. In *Proceedings of the Fifth Named Entity Workshop*, pages 10–23.

Andrew Finch, Lemao Liu, Xiaolin Wang, and Eiichiro Sumita. 2016. Target-bidirectional neural models for machine transliteration. In *Proceedings of the sixth named entity workshop*, pages 78–82.

Sittichai Jiampojamarn, Grzegorz Kondrak, and Tarek Sherif. 2007. Applying many-to-many alignments and hidden markov models to letter-to-phoneme conversion. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 372–379.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proc. ACL*.

Kevin Knight and Jonathan Graehl. 1998. Machine transliteration. *Computational linguistics*, 24(4):599–612.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the association for computational linguistics companion volume proceedings of the demo and poster sessions*, pages 177–180.

Anoop Kunchukuttan, Mitesh Khapra, Gurneet Singh, and Pushpak Bhattacharyya. 2018. Leveraging orthographic similarity for multilingual neural transliteration. *Transactions of the Association of Computational Linguistics*, 6:303–316.

Ngoc Tan Le, Fatiha Sadat, Lucie Menard, and Dien Dinh. 2019. Low-resource machine transliteration using recurrent neural networks. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, 18(2):13.

[8] https://www.meity.gov.in/
[9] http://www.cfilt.iitb.ac.in/

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational linguistics*, 29(1):19–51.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. `http://is.muni.cz/publication/884893/en`.

Bonnie Glover Stalls and Kevin Knight. 1998. Translating names and technical terms in arabic text. In *In Proceedings of the Workshop on Computational Approaches to Semitic Languages*, page 34–41.

Kaisheng Yao and Geoffrey Zweig. 2015. Sequence-to-sequence neural net models for grapheme-to-phoneme conversion. *arXiv preprint arXiv:1506.00196*.

Min Zhang, Haizhou Li, Ming Liu, and A Kumaran. 2012. Whitepaper of news 2012 shared task on machine transliteration. In *Proceedings of the 4th Named Entity Workshop*, pages 1–9. Association for Computational Linguistics.