# Statistical Romanization for Abugida Scripts: Data and Experiment on Khmer and Burmese

Chenchen Ding[1] Vichet Chea[2] Win Pa Pa[3] Masao Utiyama[1] Eiichiro Sumita[1]
[1]ASTREC, NICT, Japan [2]NIPTICT, Cambodia [3]UCSY, Myanmar

[1]{chenchen.ding, mutiyama, eiichiro.sumita}@nict.go.jp
[2]vichet.chea@niptict.edu.kh [3]winpapa@ucsy.edu.mm

## 1 Introduction

Linguistically, romanization is the task of transforming a non-Latin writing system into Latin script. It is a language-specific task in natural language processing (NLP). Despite standard romanization systems, conventional transcription methods are applied prevalently in many languages. The spellings are thus inconsistent and complicated in some cases. Consequently, statistical approaches are required for an efficient solution on this task.

We focus on the name romanization for languages using abugida scripts. Different from an alphabetic system, in an abugida system, consonant letters stand for a syllable with an implicit inherent vowel, and diacritics modify consonant letters to change or depress the inherent vowel. We designed annotation schemes to establish a precise, consistent, and monotonic correspondence between the two different writing systems on grapheme level, through which various machine learning approaches are facilitated.

Specifically, we collect and manually align $7,658$ and $2,335$ person name romanization instances for Khmer and Burmese (Myanmar), respectively. Both languages are with limited resources and NLP studies. Experimental results demonstrate that standard approaches of conditional random fields (CRF) and support vector machine (SVM) supervised by the manually annotated data achieve high precision in romanization. The annotated data have been released under a CC-BY-NC-SA license.[1]

## 2 Related Work

In engineering practice, the romanization task is a string-to-string transformation, which can be cast as a simplified translation task working on grapheme level rather than on word (or phrase) level with no (or few) reordering operations. Hence, general SMT techniques can be facilitated once training data are prepared. The phrase-based SMT plays a role of baseline in recent workshops [1, 2], whereas neural network techniques provide further gains in performance [4, 5]. Although neural network-based, pure string-to-string approaches prove powerful on different transliteration tasks, there is still room for improvement, especially on tasks between different writing systems. For example, the Thai-to-English task, which is similar to our task, has relatively poor performance in NEWS 2015. The problem around diacritics in abugida is also stated in Kunchukuttan and Bhattacharyya [6]. General techniques may offer acceptable solutions overall, but specialized investigation and processing are required for further improvement on tasks for specific languages. In the remainder of the paper, the Khmer and Burmese data with manual annotation are described in Secs. 3 and 4, respectively. Experiments are reported in Sec. 5, and Sec. 6 contains the conclusion and future work.

## 3 Khmer Data

We collect and annotate $7,658$ real Khmer name romanization instances. An example of a Khmer name aligned with its romanization is illustrated in Fig. 1.

A special feature of Khmer script is that there are two series of consonant letters, which have different inherent vowels. Furthermore, diacritics represent different vowels when added to corresponding consonant series, which leads to a very complicated vowel system. Another feature of Khmer script is that the vertically stacked consonant letters are very common. One reason is the abundant consonant clusters in phonology; another reason is the etymological spelling of Sanskrit (Pali) derived words. Those stacked letters are not strictly based on the syllable structure. Additionally (and more problematically), the virama, i.e., the diacritic used to suppress the inherent vowel, is absent in Khmer script, which makes the identification of onset and coda difficult.

[1]The Khmer data are available at http://niptict.edu.kh/khmer-name-romanization-with-alignment-on-grapheme-level/. The Burmese data are available at http://www.nlpresearch-ucsy.edu.mm/NLP_UCSY/name-db.html. Please contact the first author to obtain the data if the servers are unstable.

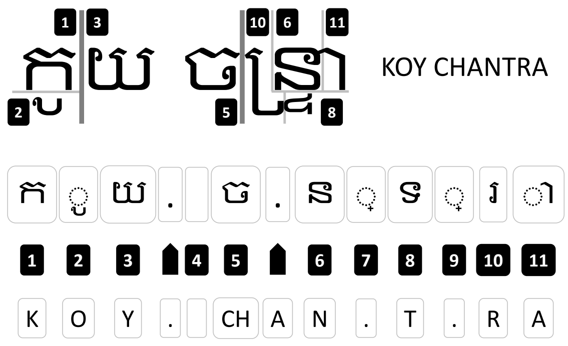


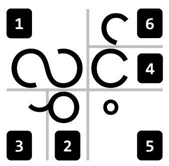
Figure 1: A Khmer name composed of two words. The upper part is the raw string pair and the lower part is the grapheme-aligned pair. In the raw Khmer string, the bold lines show the boundary of the unbreakable unit in writing and the thin lines distinguish the components. The order of Khmer letters is marked by numbers. $\boxed{4}$ is the space; $\boxed{7}$ and $\boxed{9}$ are stacking operators. The "." stands for inserted inherent vowels on the Khmer side and for a silent placeholder on the Latin side.

As phonemic syllables and writing units do not match well in Khmer script. Consistent alignment principles are thus difficult to establish if the syllables or writing units are taken as atoms in processing. Furthermore, there are numerous types of syllables and writing units, which are too complex for statistical model learning because of sparseness. Based on these facts, we prefer a pure character-level alignment, where standalone consonant letters, diacritics, and the invisible staking operator are separated and treated equally as grapheme on the Khmer side.

A consequent problem is the inherent vowels, once we completely ignore the syllable structure. We cannot judge whether a standalone consonant stands for only one consonant or contains a further inherent vowel, due to ambiguity in the writing system. We thus apply a scheme to insert a mark to represent the inherent vowel for all the "bare" consonant letters (i.e., consonant letters without any diacritics or stacking operators) to establish a consistent alignment. This insertion is thus decisive based on the surface spelling, and the ambiguity on the presence or absence of the inherent vowel is converted to whether the inserted mark is silent.

In the example of Fig. 1. the consonant letters are $\boxed{1}$, $\boxed{3}$, $\boxed{5}$, $\boxed{6}$, $\boxed{8}$, and $\boxed{10}$, where $\boxed{3}$ and $\boxed{5}$ are bare consonant letters,[2] after which the inherent vowel is inserted (noted by a dot here and indicated by a wedge without original index). Then a character-by-character alignment can be established.[3] According to our overall principles, Khmer consonant letters are aligned to Latin consonant graphemes; diacritics, including inserted inherent vowels, are aligned to Latin vowel graphemes; and any silent part is aligned to a placeholder.

## 4 Burmese Data



We collect and annotate 2,335 real Burmese name romanization instances, including students and faculty names in a university, public figure names and names from minority nationalities in Myanmar.

Syllable are clear and integrated units in Burmese script, which can be identified by rules, i.e., all diacritics and consonant letters with a virama must be attached to the letter they modify [3]. As an example, a Burmese syllable composed of six characters is illustrated in the beginning of the section, where the numbers indicate the order of the characters in the composition. $\boxed{1}$ and $\boxed{4}$ are consonant letters, and $\boxed{2}$, $\boxed{3}$, $\boxed{5}$, and $\boxed{6}$ are diacritics. Specifically, $\boxed{2}$ and $\boxed{3}$ form consonant clusters with $\boxed{1}$, $\boxed{5}$ is a tone mark, and $\boxed{6}$ is the virama to depress the inherent vowel of $\boxed{4}$ to form the coda of the syllable.

In order to provide an efficient interface for the Romanization task, we step into the syllable structure and further extract sub-syllabic segments, , i.e., onset, rhyme, and tone, to achieve a precise and consistent correspondence between Burmese and Latin scripts. Fig. 2 is a diagram of the sub-syllabic segmentation scheme. It is relatively intuitive to divide a Burmese syllable into two parts, onset and rhyme, despite the difficulties introduced by medial consonants. The rhyme is not further dividable except in to stripe tones annotated by explicit marks.

Specifically, four diacritics are used to represent the medial consonants, i.e., ***yapin***, ***yayit***, ***wahswe***, and ***hahto*** to represent /-j-/, /-j-/,[4] /-w-/, and /-$^{h}$-/,[5] respectively. As shown in Fig. 2, *yapin*, *yayit*, and *hahto* are placed into the onset part, but *wahswe* into the rhyme part. This scheme is adopted for two reasons. (1) The phonotactical constraints are strict on *yapin*, *yayit*, and *hahto*, but loose on *wahswe*. *Hahto* is only used on sonorants (nasals and approximants) and *yapin* / *yayit* only on velars and bilabials.[6] In contrast, *wahswe* can be freely combined with all consonant phonemes with the triv-

[2] $\boxed{2}$ and $\boxed{11}$ are diacritics for $\boxed{1}$ and $\boxed{10}$, respectively, and $\boxed{6}$ and $\boxed{8}$ are followed by staking operators. So these four consonant letters are not bare.

[3] As the task is to transform Khmer script to Latin script, the graphemes are not guaranteed to be single letters on the romanization side, e.g., the $\boxed{5}$ corresponds to `CH` here.

[4] *Yayit* originally represents /-r-/ while in the modern standard Burmese the phoneme /r/ has been merged into /j/.

[5] actually a voiceless sign, e.g., changing /n-/ to /n̥-/.

[6] *Yapin* can also be combined with /l-/.

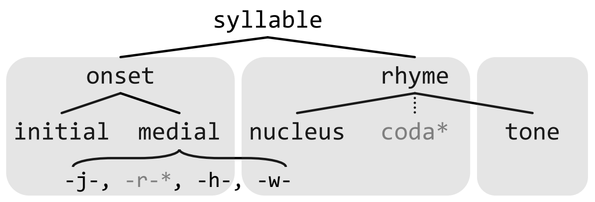


Figure 2: Burmese sub-syllabic segmentation.

ial exception of /w-/.[7] (2) *Yapin*, *yayit*, and *hahto* may change the property of the initial consonants while *wahswe* may change the property of the nucleus vowels. Besides the voiceless marker of *hahto*, *yapin / yayit* palatalize velar consonants.[8] Contrarily, *wahswe* may add a rounded feature to the following nucleus vowels.[9] The two issues affect conventional romanization spelling that multigraphs of Latin letters are used to transcribe onset clusters with *yapin*, *yayit*, and *hahto*, or *wahswe*-rhyme clusters. So, the scheme is the only feasible way to segment Burmese onset and rhyme to achieve a monotonic and one-to-one and alignment with Latin script in conventional romanization.

Burmese has two codas, nasal and glottal, and three tones, creaky, low, and high. The three tones can be combined freely for open and nasal-ended syllables, but not for glottal ended ones. The glottal ending thus may be regarded as a fourth tone in some analysis. Further, the nasal ending can be regarded as a nasalization of nucleus vowels. Hence, the coda is not a necessary component from an extreme viewpoint, which places the nasal ending into nuclei and the glottal into tones. However, in the writing system, the nasal and glottal endings are represented by consonant letters with a virama. These "coda-letters" affect nucleus vowels, so that they are not completely detachable. As a result, only two tone marks, i.e., the visarga (high) and *aukmyit* (creaky) are segmented in our scheme, as they are "pure" tone marks,[10] and any further segmentation would ruin the integrated of the rhyme.

Specific transposition is required for the sub-syllabic segmentation. An examples is illustrated in Fig. 3. For the onset-rhyme segmentation, an onset part with multiple components is coded in the order of "*C* + *yapin / yayit* + *wahswe* + *hahto*", where *C* is the initial consonant.[11] It is obvious that the segmentation cannot be conducted under the coding order once *wahswe* and *hahto* appear simultaneously. Hence, a ***wahswe-hahto* swapping** is required. For the rhyme-tone segmentation, one problem is the order of the *aukmyit* and virama in nasal-ended creaky-toned syllables.[12] As the standard order should be "*aukmyit* + virama" in coding,[13] an ***aukmyit*-virama swapping** is required.

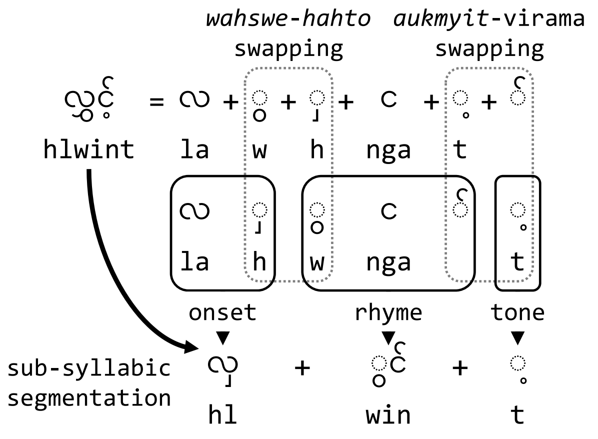


Figure 3: Transposition before segmentation.

In addition to the transposition pre-processing, a "dummy rhyme" is inserted for all the "bare onsets", identical to the insertion process applied on Khmer.

## 5 Experiment

We tested a state-of-the-art bi-directional LSTM-based RNN approach [8],[14] as well as standard CRF[15] and SVM[16] approaches on our data.

The experiments were cross-validated. The RNN handles the task in a sequence-to-sequence way on character-level, without using any *a priori* knowledge, while CRF and SVM take advantage of the designed segmentation and manual alignment. The features for CRF and SVM were tokens up to trigrams. The settings from the original paper were used for the RNN.[17] We evaluated the experimental results by two metrics: the accuracy of target token labeling (`TOK`)[18] and the accuracy of target strings (`LAT`) where BLEU [10] on Latin letters was used.

The experiments by LSTM-based RNN are conducted using an eight-fold cross-validation on our data without using manual alignment. The performance reaches .953 in terms of `LAT` on our Khmer

[7] /ʔw-/ may be argued in some references. The combination appears marginally in borrowing words and interjections.

[8] e.g., /kj-/ is actually /c-/ or /tɕ-/

[9] e.g., changing /-aʔ/ to /-ʊʔ/ and changing /-aɴ/ to /-ʊɴ/

[10] The visarga is usually not transcribed and *aukmyit* is inconsistently represented by a final `t` in Romanization.

[11] Multiple medial consonants for one initial consonant is possible while *yapin* and *yayit* cannot appear simultaneously.

[12] As mentioned, glottal endings take no tones.

[13] However, the swapped order may introduce no problem in displaying, so both orders are used in daily typing.

[14] `https://github.com/lemaoliu/Agtarbidir`

[15] **CRF++** [7, 11] at `http://taku910.github.io/crfpp/`

[16] **KyTea** [9] at `http://www.phontron.com/kytea/`

[17] Embedding size is 500, hidden unit dimension is 500, and batch size is 4. `AdaDelta` is used for optimization with a decay rate $\rho$ of 0.95 and an $\epsilon$ of $10^{-6}$.

[18] `TOK` cannot be applied to the RNN approach as the alignment is not an explicit variable.

|  | 2-fold | 4-fold | 8-fold |
|---|---|---|---|
| TOK | .987 / .988 | .988 / .989 | .989 / .990 |
| LAT | .974 / .977 | .976 / .978 | .977 / .979 |

Table 1: Results on Khmer data.

|  | 2-fold | 4-fold | 8-fold |
|---|---|---|---|
| TOK | .946 / .944 | .948 / .947 | .947 / .947 |
| LAT | .912 / .907 | .913 / .911 | .913 / .910 |

Table 2: Results on Burmese data.

data, which is an acceptable results. While RNN approach cannot performed well on our Burmese data, where `LAT` is no more than .718. We find the RNN actually generates "Burmese-styled" Latin transcriptions but inaccurate. We thus consider the performance to be reasonable and attribute the causes for the result to (1) the data size, which is still insufficient to support an RNN model (less than one third of the Khmer data), and (2) the mapping between Burmese and Latin scripts is complex, where many-to-many alignment between characters is common and is difficult to model without any heuristics.

Similar to RNN experiments, CRF and SVM experiments are also cross-validated, where the eight-, four-, and two-fold results are tested for comparison. The results by CRF and SVM on Khmer and Burmese data are listed in Tables 1 and 2, respectively. The performance of the two approaches are similar: `TOK` is around .99 and `LAT` is around .98 on Khmer data; `TOK` is around .95 and `LAT` is around .91 on Burmese data. The performances outperform those of RNN on both data sets, respectively. We conclude that the romanization task does not require features in a long distance, which RNN can model well, but precise local alignment provides useful and efficient information contributing to the performance. Therefore, we consider that it is the high quality of our annotated data, rather than a sophisticated model, that contributes more to the task.

As to engineering issues in practice, it takes **hours** to train an RNN model on several thousand instances in eight-fold cross-validation, while to train the SVM model in **KyTea** only takes **seconds**. Therefore, we consider RNN to be a superfluous approach for the Khmer romanization task.

## 6 Conclusion and Future Work

We focused on the Romanization tasks on Khmer and Burmese, from the data preparation to experiments and discussion of statistical approaches. The data prepared in this study have been released under a `CC-BY-NC-SA` license to promote the research of low-resourced language processing.

## Acknowledgements

We thank Dr. Lemao Liu for his help in the RNN experiments. We thank Mr. Kaing Hour for his help in checking the Khmer data.

## References

[1] Rafael E. Banchs, Min Zhang, Xiangyu Duan, Haizhou Li, and A. Kumaran. Report of NEWS 2015 machine transliteration shared task. In *Proc. of NEWS*, pages 10–23, 2015.

[2] Marta R. Costa-jussà. Moses-based official baseline for NEWS 2016. In *Proc. of NEWS*, pages 88–90, 2016.

[3] Chenchen Ding, Ye Kyaw Thu, Masao Utiyama, and Eiichiro Sumita. Word segmentation for Burmese (Myanmar). *ACM Transactions on Asian and Low-Resource Language Information Processing*, 15(4):22, 2016.

[4] Andrew Finch, Lemao Liu, Xiaolin Wang, and Eiichiro Sumita. Neural network transduction models in transliteration generation. In *Proc. of NEWS*, pages 61–66, 2015.

[5] Andrew Finch, Lemao Liu, Xiaolin Wang, and Eiichiro Sumita. Target-bidirectional neural models for machine transliteration. In *Proc. of NEWS*, pages 78–82, 2016.

[6] Anoop Kunchukuttan and Pushpak Bhattacharyya. Data representation methods and use of mined corpora for Indian language transliteration. In *Proc. of NEWS*, pages 78–82, 2015.

[7] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML*, pages 282–289, 2001.

[8] Lemao Liu, Andrew Finch, Masao Utiyama, and Eiichiro Sumita. Agreement on target-bidirectional LSTMs for sequence-to-sequence learning. In *Proc. of AAAI*, pages 2630–2637, 2016.

[9] Graham Neubig, Yosuke Nakata, and Shinsuke Mori. Pointwise prediction for robust, adaptable Japanese morphological analysis. In *Proc. of ACL-HLT*, pages 529–533, 2011.

[10] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: A method for automatic evaluation of machine translation. In *Proc. of ACL*, pages 311–318, 2002.

[11] Fei Sha and Fernando Pereira. Shallow parsing with conditional random fields. In *Proc. of HLT-NAACT*, pages 134–141, 2003.