Statistical Khmer Name Romanization

Chenchen Ding[1(✉)], Vichet Chea[2], Masao Utiyama[1], Eiichiro Sumita[1], Sethserey Sam[2], and Sopheap Seng[2]

[1] Advanced Translation Technology Laboratory, ASTREC, National Institute of Information and Communications Technology, 3-5 Hikaridai, Seikacho, Sorakugun, Kyoto 619-0289, Japan
{chenchen.ding,mutiyamam,eiichiro.sumita}@nict.go.jp
[2] Research and Development Center, National Institute of Posts, Telecommunication and ICT, #41 Russian Federation Blvd., Phnom Penh, Cambodia
{vichet.chea,sethserey.sam,sopheap.seng}@niptict.edu.kh

Abstract. We discuss and solve the task of Khmer name Romanization. Although several standard Romanization systems exist for Khmer, conventional transcription methods are applied prevalently in practice. These are inconsistent and complicated in some cases, due to unstable phonemic, orthographic, and etymological principles. Consequently, statistical approaches are required for the task. We collect and manually align 7,658 Khmer name Romanization instances. The alignment scheme is designed to reach a precise, consistent, and monotonic correspondence between the two different writing systems on grapheme level, through which various machine learning approaches are facilitated. Experimental results demonstrate that standard approaches of conditional random fields and support vector machine supervised by the manual alignment achieve a precision of .99 on grapheme level, which outperforms a state-of-the-art recurrent neural network approach in a pure sequence-to-sequence manner. The manually aligned data have been released under a license of CC BY-NC-SA for the research community.

1 Introduction

Romanization is a linguistic task used to transform a non-Latin writing system into Latin script—which is not a huge but an important and language-specific task in natural language processing (NLP), especially in statistical machine translation (SMT)—for those languages that do not use Latin script in orthography. In academia, orthographic transliteration, phonemic transcription, or mixed systems with certain trade-off have been developed for different languages in various studies. In daily life, however, casual and conventional ways are more commonly used in many languages, with varying inconsistency. Chinese (Mandarin) is a typical example of a language with a stable Romanization system, *pinyin*, which is used officially and in daily life. As to the case of Japanese, the *Hepburn Romanization* is the most common system for daily use, including passports, although certain variants in notation (e.g., around long vowels) are

K. Hasida and W. P. Pa (Eds.): PACLING 2017, CCIS 781, pp. 179–190, 2018.
https://doi.org/10.1007/978-981-10-8438-6_15

allowed. Korean is a case with considerable variants in Romanization, where conventional or personalized spellings are prevalent.

In this study, we focus on the name Romanization of Khmer, a Southeast Asian language with limited NLP studies. Similar to the languages mentioned above, several Romanization systems have been designed for Khmer. However, the case of Khmer is most similar to that of Korean, where Romanization with conventional spellings is strongly preferable in practice. Compared with Korean, Khmer has a more complicated phonology, less clear syllable structures, and more etymologically oriented spellings, all of which make the task more problematic. Hence, statistical approaches based on real data are required to solve the problem. In this paper, we provide satisfactory solutions for the Khmer name Romanization task with detailed descriptions on data preparation, statistical approaches, and discussions. We believe this is the first comprehensive work for the specific task and we have released the data prepared and used in this study under a license of `CC BY-NC-SA`.[1]

Specifically, we collect 7,658 Romanization instances from real Khmer person names. We then design an alignment scheme to manually annotate the Romanization instances. The scheme is carefully designed to realize a (1) consistent, (2) precise, and (3) monotonic alignment on grapheme level. In practice, the alignment is conducted manually at the very beginning, and automatic cross-checking with manual modification is conducted repeatedly to clean up mistakes. Finally, difficult and unusual instances are picked up and checked with further discussions between annotators. The elaborated manual alignment on a dataset with a considerable size thus provides a solid foundation for further investigation.

In the experiments, we first test rule-based approaches, which do not require training data but use manual rules. The performance is mediocre. Ad-hoc rules only provide insignificant gains. For statistical approaches, two standard machine learning methods, *conditional random fields* (CRF) and *support vector machine* (SVM), are tested. Because of the well-prepared grapheme-level alignment, the Romanization task is simplified to be a sequence labeling task. The performance is satisfactory: the precision on grapheme level reached .99 in cross-validation. We also conduct a direct sequence-to-sequence experiment without using manual alignment but applying a state-of-the-art bidirectional long short-term memory (LSTM) based recurrent neural network (RNN) approach. The performance is good, but does not match CRF's and SVM's results, which take advantage of manual alignment. Therefore, We consider that it is the high quality of our manually aligned data, rather than a sophisticated model, that contributes more to the task.

The remainder of the paper is organized as follows: In Sect. 2, we introduce the background of Khmer and related work on Romanization in NLP. In Sect. 3, we prove descriptions of the alignment scheme we designed and applied to annotate the data. Section 4 reports the evaluation and discussion based on experiments, and Sect. 5 provides a conclusion.

[1] http://niptict.edu.kh/khmer-name-romanization-with-alignment-on-grapheme-level/.

2 Background

Compared with the other Southeast Asian languages (e.g., Burmese and Thai), the phonology of Khmer has two obvious features. First, Khmer is **not** a tonal language, which is quite unusual in Southeast Asia. To compensate for the absence of tones, Khmer has a large set of vowel phonemes with a size of up to (at least) 31. Second, Khmer has abundant types of consonant clusters. A two-consonant cluster in a syllable's onset is common (three at the maximum). Phonotactic constraints are loose in Khmer, where complex consonant clusters can appear at the beginning of a syllable. A complete list may contain up to around 80 types of consonant combinations.

The two phonemic features of Khmer are deeply related to Khmer's abugida writing system. In a general abugida system, each standalone consonant letter can form a complete syllable with a hidden inherent vowel. The inherent vowel can be changed to other vowels or suppressed using various diacritic marks. The special feature of Khmer script is that there are two series (or registers) of consonant letters, which have different inherent vowels. The two series are usually mentioned as **a**-series (1st-series) and **o**-series (2nd-series). Furthermore, diacritics represent two vowels when added to corresponding consonant series.[2] which leads to a very complicated vowel system. Another feature of Khmer script is that the stacked consonants are very common. One reason is the abundant consonant clusters in phonology; another reason is the etymological spelling of Sanskrit (Pali) derived words. The stacked consonants 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,[3] which makes the identification of onset and coda difficult.

Further introduction of Khmer's phonology and script can be referred to in the following resources available on-line: a sketchy one [3] and a detailed one [6]. We provide examples in Figs. 1 and 2 for illustration. In the first example, three consonant letters, ❷, ❹, and ❻ are stacked to a block to represent a consonant cluster and a dependent vowel diacritic ❼ is attached to the block to form an unbreakable unit in writing. However, the first consonant ❷ in this unit is the coda of the first syllable (❶❷) and the left ❹ and ❻ are the onset of the second syllable (❹❻❼❽). This is a typical instance of inconsistency between phonemic analysis and writing systems. Furthermore, the first and third units, both of which are composed of a single consonant letter (❶ and ❽, respectively), play different roles. The first unit stands for the onset and nucleus of the first syllable with the inherent vowel of letter ❶. The third unit is the coda of the second syllable, where the inherent vowel of letter ❽ is suppressed, without any explicit notation in writing. The second example has a more consistent and

[2] Some diacritics always stand for one vowel. There are also diacritics serving as "shifter" to switch the series of certain consonant letters, which affect the vowel sound of other diacritics.

[3] Actually, Khmer script has this diacritic, called *viriam*, but uses obsoletely.

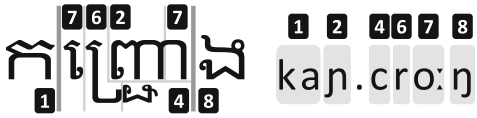

Fig. 1. Khmer word with the meaning of *fox*. The bold lines on the Khmer word show the boundary of the unbreakable unit in writing and the thin lines distinguish the components. The order of Khmer letters is marked by numbers. The missing 3 and 5 are invisible stacking operators for generating the unit of 2 4 6. The 7 is a separated diacritic with two parts. IPA for the whole word and each letter is shown.

Fig. 2. Khmer word with the meaning of *teacher*, borrowed from Sanskrit *ācārya*. Notation in this figure is identical to that in Fig. 1. The missing 6 is the invisible stacking operator for generating the unit of 5 7. The third unbreakable unit 5 7 is completely silent in pronouncing, but only written in orthography due to the etymology.

Fig. 3. A Khmer name composed of two words. The left side is the raw string pair and the right side is the grapheme-aligned pair. 4 is the space; 7 and 9 are stacking operators. The "." stands for inserted inherent vowels on the Khmer side (annotated by a hedge) and for a silent placeholder on the Latin side.

clearer match between writing and pronouncing, in that the first and second units (1 2 and 3 4, respectively) exactly correspond to the two syllables. The problem here is the third unit (5 6), which is totally silent, but appears in writing only for the etymological reason. These examples illustrate the difficulties in even a pure phonemic transcription for Khmer script, where orthographic and etymological facts are involved. As for Romanization in practice, the task further suffers variants in mapping from phoneme to grapheme, as noted by the Latin alphabet.

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.[4] The problem around diacritics in abugida is also stated in [7]. General techniques may offer acceptable solutions overall, but specialized investigation and processing are required for further improvement on tasks for specific languages, or language pairs.

3 Data

3.1 Collection

We collect 7,658 real Khmer names with corresponding Romanization in pairs. Khmer names usually consist of two words, a family name coming first, followed by a given name, separated by a space in writing. However, the family name is not an obliged element, i.e., a person may have no family name but only a given name. On the other hand, a given name may contain more than one word. As a result, for a name containing two words, it is not always clear that it is a family-given name pair or a two-word given name. We do not explicitly distinguish the family name and given name in our data. The spaces in names are treated as an ordinary character and kept constant in Romanization. That is, Romanization is always word-by-word, without any merging or splitting operations. An example of a two-word Khmer name with its Romanization is illustrated on the left side of Fig. 3.

3.2 Overall Principles in Alignment

As mentioned, phonemic syllables and unbreakable writing units do not match well in Khmer script. Consistent alignment principles are thus difficult to establish if the syllables or units are taken as atoms in processing. Furthermore, there are numerous types of syllables and writing units that may contain up to four phonemes, 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 in a character-based alignment. 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

[4] The original names are western names in the Thai-to-English task, which may make the grapheme correspondence more varied and inconsistent.

vowel is converted to whether the inserted mark is silent.[5] As a result, the problem is treated as uniformly as the other silent characters.

The right side of Fig. 3 illustrates an alignment example. The consonant letters here are 1, 3, 5, 6, 8, and 10, where 3 and 5 are bare consonant letters,[6] 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.[7] According to the 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.[8] Further alignment details caused by specific use is described in the following subsection.

In Fig. 3 as well as in our practice, we use the same mark for the Khmer side inserted inherent vowel and the silent placeholder on the Romanization side. This introduces no confusion because the mark is decisively inserted on the Khmer side and decisively deleted in the Romanization results.

3.3 Special Cases in Alignment

The alignment under our overall principles is almost monotonic and one-to-one. However, there are two specific cases to be discussed: (1) multiple diacritics for one consonant letter and (2) stacked consonants letters for a single phoneme.

The first case is mainly caused by the *anusvara* (adding nasal ending), *visarga* (adding aspiration ending), and *series shifters* changing the a-/o-series of consonant letters. Figure 4 illustrates the examples. When a consonant letter takes an anusvara directly, it is not bare anymore, so the Latin letters for inherent vowel and the nasal ending (commonly transcribed into `M`) are taken as one grapheme aligned to the anusvara. The anusvara can, however, modify other diacritics to add a nasal ending, where only the `M` endings are aligned to anusvara and the vowel value is taken by the preceding diacritics. The first two instances in Fig. 4 show the difference in alignment on anusvara. The alignment around visarga, which is commonly transcribed into `S`, is identical to that around anusvara. The middle two instances in Fig. 4 show the alignment around the series shifter, which is similar to the anusvara (visarga), but inserted between a consonant letter and other diacritics. The shifter is actually modifying the preceding consonant letter, so the value of the vowel will be afforded by other diacritics if there is any. The shifter itself has no specific value unless it is the only diacritic of a consonant.

[5] Generally, the diacritics in an abugida system are observed as vowel shifters to change the inherent vowel. From the viewpoint of this study, the diacritics are actually treated as vowel notes and the inherent vowel is treated as one with a zero-alternant form, which the insertion processing makes explicit.

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

[7] 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 5 corresponds to `CH` here.

[8] The stacking operator is always aligned to a silent placeholder.

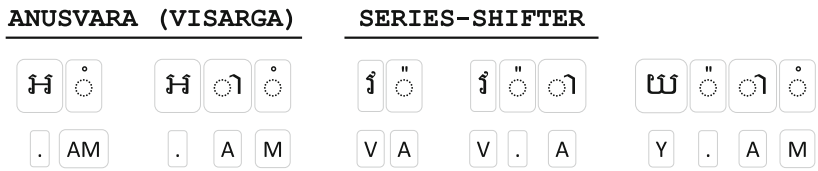

Fig. 4. Alignment principles around anusvara (visarga), series shifters, and an alignment example for a Chinese-style name with both diacritics.

Fig. 5. Alignment examples of two western-style names, where the consonants `Z` and `F` are non-native phonemes of Khmer. The original value (common transcription) of consonant letters combined to represent the two sounds is placed upward.

The rightmost example in Fig. 4 is a real name taking both anusvara and the shifter simultaneously, aligned according to our principles.

The second case is caused by certain non-native consonants in Khmer, mainly appearing in loanwords, and here, appearing in westernized names. Figure 5 provides two real examples. It can be observed that `Z` is represented by stacked Khmer letters for `H` and `S`, and `F` is represented by stacked Khmer letters for `H` and `V`. The consonant letter of `H` is commonly stacked over another consonant letter of an approximate pronunciation to represent a non-native consonant. Therefore, the corresponding Latin consonant graphemes are aligned to the second Khmer letters and the first Khmer letter `H` is aligned to a silent placeholder. In Fig. 5 (on the right), the stacked consonants for `F` are further modified by two diacritics, where the first is a series shifter.

As illustrated in this subsection, the combination of consonant letters and various diacritics is complicated in Khmer script, and the grapheme-level alignment we designed can cover different cases consistently, to offer a monotonic and one-to-one alignment. However, the Romanization of Khmer characters is not totally position-free under our alignment scheme, and at least a window of tri-gram on Khmer characters is required to provide enough information for a correct Romanization.

4 Experiment

4.1 Questions and Approaches

The well-aligned data prepared in this study have already provided a solid foundation of the Romanization task. Therefore, we investigate the following two

questions through experiments: (1) to what extent can statistical approaches help the task and (2) to what extent can our manual alignment help statistical approaches.

For the first question, we first try a rule-based transcription to investigate the performance. The rules used are edited manually, based on conventional spellings in Romanization rather than academic standards. Then we try a state-of-the-art neural network-based sequence-to-sequence approach trained on surface string pairs in our data, without using the alignment. The neural network outperforms the rule-based approach by a large margin. The results illustrate the necessity of real data and statistical approaches in this task.

For the second question, we further try two standard and widely used machine learning approaches, CRF and SVM, trained on the grapheme-level alignment. Because the alignment has cleaned up two difficult problems in transformation, i.e., source-side segmentation (actually, character-by-character) and source-to-target alignment, the task is converted to a sequence labeling task, to which these compact and efficient approaches are applicable. The results yielded by the two approaches outperform the neural network's results, demonstrating that the manual alignment does provide useful information and boosts the performance efficiently.

The details of the experiments are described in the following subsections. Experimental results are evaluated in two ways: the accuracy on source-to-target grapheme transcription (`GRAPH`) and the accuracy on target strings (`LATIN`). `GRAPH` is based on grapheme-level alignment, where all Khmer letters, including the inserted inherent vowel, are counted in the calculation.[9] `GRAPH` cannot be applied to the sequence-to-sequence experiments as the alignment is not an explicit variable in the processing and final results. For `LATIN`, we simply apply the BLEU score [11], the most common measure in SMT, on alphabet level, where silent placeholders in Romanization are deleted before calculation.[10]

4.2 Rule-Based Transcription

Although the Romanization of Khmer is not consistent, there is preference for each letter. Figures 6 and 7 show the most preferred transcription of Khmer letters. A naïve mapping based on the listed transcriptions cannot reach a satisfactory performance. The accuracy is .746 on `GRAPH` and .515 on `LATIN`. When we focus more on some conventional spelling features, so as to double the consonant letters after short vowels or to transform specific stacking consonants, (e.g., the case in Fig. 5), we have a better result, a `GRAPH` of .909 and a `LATIN` of .821. Further improvement is difficult. We try ad-hoc rules in an exhaustive way and find the upper boundary of the rule-based mapping to be around .95 for

[9] Spaces are not counted because they are always maintained constant. Stacking operators as well as other silent Khmer letters are counted.

[10] Spaces are taken as one character in the BLEU calculation.

GRAPH and .88 for LATIN. Although the performance is not perfect, we consider it reasonable and acceptable in terms of simplicity.[11]

4.3 Sequence-to-Sequence Transliteration

In recent research, direct sequence-to-sequence approaches launched by neural network techniques have been widely applied in various NLP tasks. We experiment a state-of-the-art LSTM-based RNN approach with a bidirectional search in decoding[12] [9]. The approach performs well on different transliteration tasks.

The experiment is conducted using an eight-fold cross-validation on our data without using manual alignment. The entire data are split into eight parts, with each part taken as a test set and the left parts used for training. As separated development data are needed for tuning the iteration times, one-thirtieth of the training data is sampled. Other hyper-parameters are based according to the original paper: embedding size is 500, hidden unit dimension is 500, and batch size is 4. AdaDelta is used for optimization with a decay rate ρ of 0.95 and an ϵ of 10^{-6}.

The result of LSTM-based RNN reaches .953 in terms of LATIN. Compared with the rule-based mapping result, a machine learning approach clearly performs better on the task, even without applying any specific *a priori* knowledge. The comparison of the two approaches provides a clear and solid answer to the first question.

4.4 Alignment-Based Sequence Labeling

We test two standard machine learning approaches, CRF and SVM, by handling the task in a sequence labeling manner. That is, the Latin graphemes (including the silent placeholder) are tried as labels for each Khmer character.

We use the **CRF++** toolkit[13] [8,12] and the **KyTea** toolkit[14] [10] for CRF and SVM experiments, respectively. Both toolkits are open-sourced. Similar to RNN experiments, CRF and SVM experiments are also cross-validated, where the eight-, four-, and two-fold results are tested for comparison. As to the settings of the two approaches, we basically use up to tri-gram on Khmer characters as input features. As mentioned, this is the minimum size of a window to provide adequate information under our alignment scheme. Specifically, the `-charn` and the `-charw` options are set to three for **KyTea**. The features used for **CRF++** are C_n^{n+k} ($k \in [1,2], n \in [-k,0]$) for character sequences and C_n ($n \in [-2,2]$) for single characters. As the training speed of **KyTea** is very fast under this setting, we ultimately try an exhaustive leave-one-out experiment, in which each instance is left for testing and all the rest instances are used for training. Table 1 lists the evaluation on the whole dataset.

[11] A Python implementation for the rule-based transcription with different layers of rules is available at http://www2.nict.go.jp/astrec-att/member/mutiyama/software.html.

[12] An open-sourced tool is available at https://github.com/lemaoliu/Agtarbidir.

[13] http://taku910.github.io/crfpp/.

[14] http://www.phontron.com/kytea/.

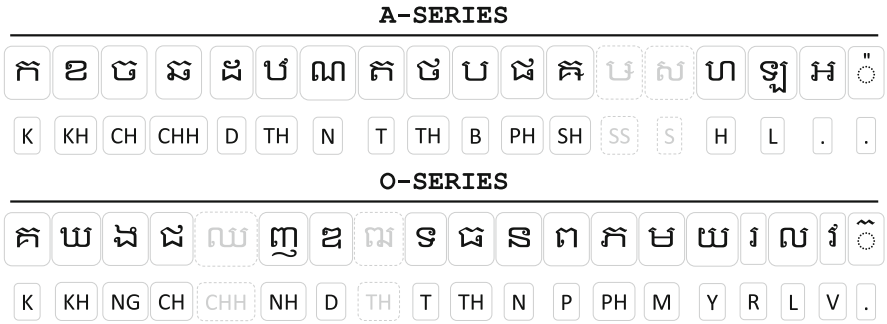

Fig. 6. Preferable transcription for Khmer consonant letters. A-series and o-series letters are listed. The gray letters are rarely used and never appear in our data. The final diacritics in the two series are the two series shifters, which change the consonant letters in the other series to their own series.

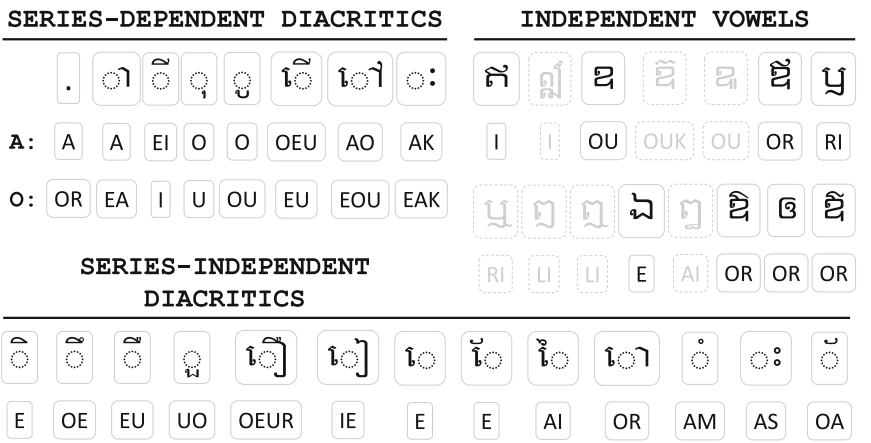

Fig. 7. Preferable transcription for different Khmer vowel diacritics and letters. Diacritics with consonant series-dependent transcriptions are listed on the upper left, a-series, and o-series respectively. Diacritics with identical transcriptions are listed at the bottom. These diacritics may stand for different vowels for consonants from different series, without being reflected in Romanization. In the upper right, standalone letters for vowels are listed. Rare vowels not appearing in our data are marked in gray.

Table 1. Evaluation results (`GRAPH/LATIN`) of CRF and SVM in cross-validation.

	2-fold	4-fold	8-fold	leave-1-out
`CRF`	.987/.974	.988/.976	.989/.977	—
`SVM`	.988/.977	.989/.978	.990/.979	.990/.980

The performance of the two approaches is nearly identical. The performance on `GRAPH` is around .99 and on `LATIN` is around .98, which outperforms the RNN's .95. The results is satisfactory and reasonable. As to the answer to the second question, we conclude that the Khmer 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.

As to engineering issues in practice, it takes **hours** to train an RNN model on more than 6,000 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. As Table 1 shows, two-, four-, eight-fold, and leave-one-out results actually do not differ much, which indicates that several thousand instances are sufficient for the model training.

5 Conclusion

In this paper, we focus on the task of Khmer name Romanization, a task not huge, but with its own difficulties. By collecting and elaborately aligning more than 7,000 real instances, we provide a solid foundation for the task. Experimental results demonstrate that a rule-based approach is not sufficient to solve the Romanization task, while statistical machine learning approaches trained on our manual alignment can achieve an accuracy of .99 on the grapheme level. Therefore, we believe that the Khmer name Romanization task has actually been solved, provided our data and standard machine learning techniques. As our manually aligned dataset plays a key role in solving the task, we release it under a license of `CC BY-NC-SA` for the research community.

References

1. Banchs, R.E., Zhang, M., Duan, X., Li, H., Kumaran, A.: Report of NEWS 2015 machine transliteration shared task. In: Proceedings of NEWS, pp. 10–23 (2015)
2. Costa-jussà, M.R.: Moses-based official baseline for NEWS 2016. In: Proceedings of NEWS, pp. 88–90 (2016)
3. Ehrman, M.E., Sos, K., Kheang, L.H.: Contemporary Cambodian – grammatical sketch (1974). https://www.livelingua.com/fsi/Fsi-ContemporaryCambodian-GrammaticalSketch.pdf
4. Finch, A., Liu, L., Wang, X., Sumita, E.: Neural network transduction models in transliteration generation. In: Proceedings of NEWS, pp. 61–66 (2015)
5. Finch, A., Liu, L., Wang, X., Sumita, E.: Target-bidirectional neural models for machine transliteration. In: Proceedings of NEWS, pp. 78–82 (2016)
6. Huffman, F.E.: Cambodian system of writing and beginning reader with drills and glossary (1970). http://www.pratyeka.org/csw/hlp-csw.pdf
7. Kunchukuttan, A., Bhattacharyya, P.: Data representation methods and use of mined corpora for Indian language transliteration. In: Proceedings of NEWS, pp. 78–82 (2015)
8. Lafferty, J., McCallum, A., Pereira, F.: Conditional random fields: probabilistic models for segmenting and labeling sequence data. In: Proceedings of ICML, pp. 282–289 (2001)

9. Liu, L., Finch, A., Utiyama, M., Sumita, E.: Agreement on target-bidirectional LSTMs for sequence-to-sequence learning. In: Proceedings of AAAI, pp. 2630–2637 (2016)
10. Neubig, G., Nakata, Y., Mori, S.: Pointwise prediction for robust, adaptable Japanese morphological analysis. In: Proceedings of ACL-HLT, pp. 529–533 (2011)
11. Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: BLEU: a method for automatic evaluation of machine translation. In: Proceedings of ACL, pp. 311–318 (2002)
12. Sha, F., Pereira, F.: Shallow parsing with conditional random fields. In: Proceedings of HLT-NAACL, pp. 134–141 (2003)