Simplified Abugidas

Chenchen Ding, Masao Utiyama, and Eiichiro Sumita
Advanced Translation Technology Laboratory,
Advanced Speech Translation Research and Development Promotion Center,
National Institute of Information and Communications Technology
3-5 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-0289, Japan
{chenchen.ding, mutiyama, eiichiro.sumita}@nict.go.jp

Abstract

An abugida is a writing system where the consonant letters represent syllables with a default vowel and other vowels are denoted by diacritics. We investigate the feasibility of recovering the original text written in an abugida after omitting subordinate diacritics and merging consonant letters with similar phonetic values. This is crucial for developing more efficient input methods by reducing the complexity in abugidas. Four abugidas in the southern Brahmic family, i.e., Thai, Burmese, Khmer, and Lao, were studied using a newswire $20,000$-sentence dataset. We compared the recovery performance of a support vector machine and an LSTM-based recurrent neural network, finding that the abugida graphemes could be recovered with 94% – 97% accuracy at the top-1 level and 98% – 99% at the top-4 level, even after omitting most diacritics (10 – 30 types) and merging the remaining 30 – 50 characters into 21 graphemes.

1 Introduction

Writing systems are used to record utterances in a wide range of languages and can be organized into the hierarchy shown in Fig. 1. The symbols in a writing system generally represent either speech sounds (phonograms) or semantic units (logograms). Phonograms can be either segmental or syllabic, with segmental systems being more phonetic because they use separate symbols (i.e., letters) to represent consonants and vowels. Segmental systems can be further subdivided depending on their representation of vowels. Alphabets (e.g., the Latin, Cyrillic, and Greek scripts) are the most common and treat vowel and consonant letters equally. In contrast, abjads (e.g., the Arabic and Hebrew scripts) do not write most vowels explicitly. The third type, abugidas, also called alphasyllabary, includes features from both segmental and syllabic systems. In abugidas, consonant letters represent syllables with a default vowel, and other vowels are denoted by diacritics. Abugidas thus denote vowels less explicitly than alphabets but more explicitly than abjads, while being less phonetic than alphabets, but more phonetic than syllabaries. Since abugidas combine segmental and syllabic systems, they typically have more symbols than conventional alphabets.

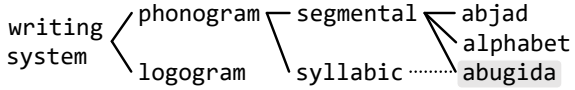

Figure 1: Hierarchy of writing systems.

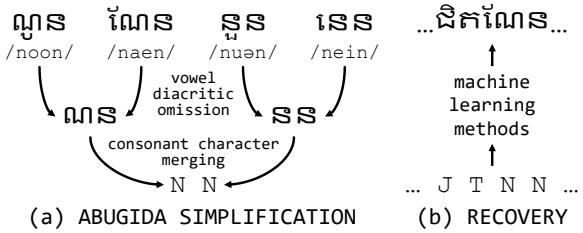

Figure 2: Overview of the approach in this study.

In this study, we investigate how to simplify and recover abugidas, with the aim of developing a more efficient method of encoding abugidas for input. Alphabets generally do not have a large set of symbols, making them easy to map to a traditional keyboard, and logogram and syllabic systems need specially designed input methods because of their large variety of symbols. Traditional input methods for abugidas are similar to those for alphabets, mapping two or three different symbols onto each key and requiring users to type each character and diacritic exactly. In contrast, we are able to substantially simplify inputting abugidas by encoding them in a *lossy* (or "*fuzzy*") way.

Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Short Papers), pages 491–495
Melbourne, Australia, July 15 - 20, 2018.

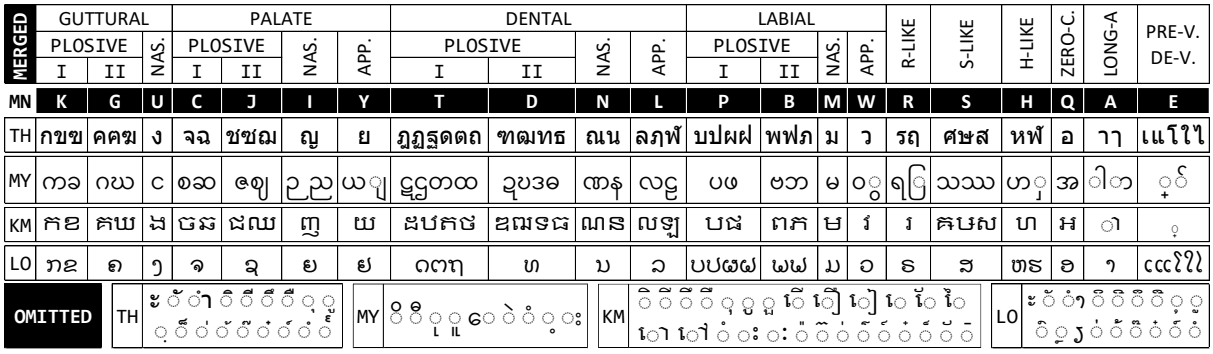

Figure 3: Merging and omission for Thai (TH), Burmese (MY), Khmer (KM), and Lao (LO) scripts. The MN row lists the mnemonics assigned to graphemes in our experiment. In this study, the mnemonics can be assigned arbitrarily, and we selected Latin letters related to the real pronunciation wherever possible.

Fig. 2 gives an overview of this study, showing examples in Khmer. We simplify abugidas by omitting vowel diacritics and merging consonant letters with identical or similar phonetic values, as shown in (a). This simplification is intuitive, both orthographically and phonetically. To resolve the ambiguities introduced by the simplification, we use data-driven methods to recover the original texts, as shown in (b). We conducted experiments on four southern Brahmic scripts, i.e., Thai, Burmese, Khmer, and Lao scripts, with a unified framework, using data from the Asian Language Treebank (ALT) (Riza et al., 2016). The experiments show that the abugidas can be recovered satisfactorily by a recurrent neural network (RNN) using long short-term memory (LSTM) units, even when nearly all of the diacritics (10 – 30 types) have been omitted and the remaining 30 – 50 characters have been merged into 21 graphemes. Thai gave the best performance, with 97% top-1 accuracy for graphemes and over 99% top-4 accuracy. Lao, which gave the worst performance, still achieved the top-1 and top-4 accuracies of around 94% and 98%, respectively. The Burmese and Khmer results, which lay in-between the other two, were also investigated by manual evaluation.

2 Related Work

Some optimized keyboard layout have been proposed for specific abugidas (Ouk et al., 2008). Most studies on input methods have focused on Chinese and Japanese characters, where thousands of symbols need to be encoded and recovered. For Chinese characters, Chen and Lee (2000) made an early attempt to apply statistical methods to sentence-level processing, using a hidden Markov model. Others have examined max-entropy models, support vector machines (SVMs), conditional random fields (CRFs), and machine translation techniques (Wang et al., 2006; Jiang et al., 2007; Li et al., 2009; Yang et al., 2012). Similar methods have also been developed for character conversion in Japanese (Tokunaga et al., 2011). This study takes a similar approach to the research on Chinese and Japanese, transforming a less informative encoding into strings in a natural and redundant writing system. Furthermore, our study can be considered as a specific lossy compression scheme on abugida textual data. Unlike images or audio, the lossy text compression has received little attention as it may cause difficulties with reading (Witten et al., 1994). However, we handle this issue within an input method framework, where the simplified encoding is not read directly.

3 Simplified Abugidas

We designed simplification schemes for several different scripts within a unified framework based on phonetics and conventional usages, without considering many language specific features. Our primary aim was to investigate the feasibility of reducing the complexity of abugidas and to establish methods of recovering the texts. We will consider language-specific optimization in a future work, via both data- and user-driven studies.

The simplification scheme is shown in Fig. 3.[1] Generally, the merges are based on the common distribution of consonant phonemes in most natural languages, as well as the etymology of the characters in each abugida. Specifically, three or four

[1]Each script also includes native punctuation marks, digit notes, and standalone vowel characters that are not represented by diacritics. These characters were kept in the experimental texts but not evaluated in the final results, as the usage of these characters is trivial. In addition, white spaces, Latin letters, Arabic digits, and non-native punctuation marks were normalized into placeholders in the experiments, and were also excluded from evaluation.

graphemes are preserved for the different articulation locations (i.e., guttural, palate, dental, and labial), that two for plosives, one for nasal (NAS.), and one for approximant (APP.) if present. Additional consonants such as trills (R-LIKE), fricatives (S-/H-LIKE), and empty (ZERO-C.) are also assigned their own graphemes. Although the simplification omits most diacritics, three types are retained, i.e., one basic mark common to nearly all Brahmic abugidas (LONG-A), the preposed vowels in Thai and Lao (PRE-V.), and the vowel-depressors (and/or consonant-stackers) in Burmese and Khmer (DE-V.). We assigned graphemes to these because we found they informed the spelling and were intuitive when typing. The net result was the omission of 18 types of diacritics in Thai, 9 in Burmese, 27 in Khmer, and 18 in Lao, and the merging of the remaining 53 types of characters in Thai, 43 in Burmese, 37 in Khmer, and 33 in Lao, into a unified set of 21 graphemes. The simplification thus substantially reduces the number of graphemes, and represents a straightforward benchmark for further language-specific refinement to build on.

4 Recovery Methods

The recovery process can be formalized as a sequential labeling task, that takes the simplified encoding as input, and outputs the writing units, composed of merged and omitted character(s) in the original abugidas, corresponding to each simplified grapheme. Although structured learning methods such as CRF (Lafferty et al., 2001) have been widely used, we found that searching for the label sequences in the output space was too costly, because of the number of labels to be recovered.[2] Instead, we adopted non-structured point-wise prediction methods using a linear SVM (Cortes and Vapnik, 1995) and an LSTM-based RNN (Hochreiter and Schmidhuber, 1997).

Fig. 4 shows the overall structure of the RNN. After many experimentations, a general "shallow and broad" configuration was adopted. Specifically, simplified grapheme bi-grams are first embedded into 128-dimensional vectors[3] and then encoded in one layer of a bi-directional LSTM,

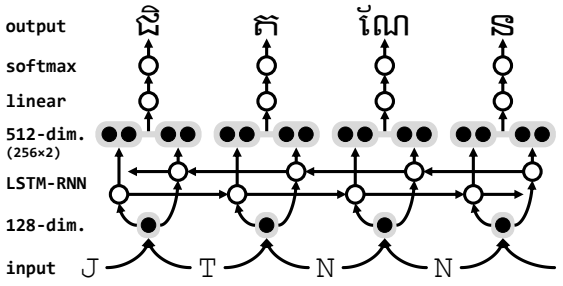

Figure 4: Structure of the RNN used in this study.

resulting in a final representation consisting of a 512-dimensional vector that concatenates two 256-dimensional vectors from the two directions. The number of dimensions used here is large because we found that higher-dimensional vectors were more effective than the deeper structures for this task, as memory capacity was more important than classification ability. For the same reason, the representations obtained from the LSTM layer are transformed linearly before the softmax function is applied, as we found that non-linear transformations, which are commonly used for final classification, did not help for this task.

5 Experiments and Evaluation

We used raw textual data from the ALT,[4] comprising around 20,000 sentences translated from English. The data were divided into training, development, and test sets as specified by the project.[5]

For the SVM experiments, we used the off-the-shelf LIBLINEAR library (Fan et al., 2008) wrapped by the KyTea toolkit.[6] Table 1 gives the recovery accuracies, demonstrating that recovery is not a difficult classification task, given well represented contextual features. In general, using up to 5-gram features before/after the simplified grapheme yielded the best results for the baseline, except with Burmese, where 7-gram features brought a small additional improvement. Because Burmese texts use relatively more spaces than the other three scripts, longer contexts help more. Meanwhile, Lao produced the worst results, possibly because the omission and merging process was harsh: Lao is the most phonetic of the four scripts, with the least redundant spellings.

The LSTM-based RNN was implemented using DyNet (Neubig et al., 2017), and it was trained using Adam (Kingma and Ba, 2014) with an initial

[2] One consonant character can be modified by multiple diacritics. In the ALT data used in this study, there are around 600 – 900 types of writing units in each script, and there could be over 1,000 on larger textual data.

[3] Directly embedding uni-grams (single graphemes) did not give good performance in our preliminary experiments.

[4] http://www2.nict.go.jp/astrec-att/member/mutiyama/ALT/

[5] Around 18,000, 1,000, and 1,000 sentences, resp.

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

learning rate of 10^{-3}. If the accuracy decreased on the development set, the learning rate was halved, and learning was terminated when there was no improvement on the development set for three iterations. We did not use dropout (Srivastava et al., 2014) but instead a voting ensemble over a set of differently initialized models trained in parallel, which is both more effective and faster.

As shown in Table 2, the RNN outperformed SVM on all scripts in terms of top-1 accuracy. A more lenient evaluation, i.e., top-n accuracy, showed a satisfactory coverage of around 98% (Khmer and Lao) to 99% (Thai and Burmese) considering only the top four results. Fig. 5 shows the effect of changing the size of the training dataset by repeatedly halving it until it was one-eighth of its original size, demonstrating that the RNN outperformed SVM regardless of training data size. The LSTM-based RNN should thus be a substantially better solution than the SVM for this task.

We also investigated Burmese and Khmer further using manual evaluation. The results of $\text{RNN}^{@1}_{\oplus 16}$ in Table 2 were evaluated by native speakers, who examined the output writing units corresponding to each input simplified grapheme and classified the errors using four levels: 0) acceptable, i.e., alternative spelling, 1) clear and easy to identify the correct result, 2) confusing but possible to identify the correct result, and 3) incomprehensible. Table 3 shows the error distribution. For Burmese, most of the errors are at levels 1 and 2, and Khmer has a wider distribution. For both scripts, around 50% of the errors are serious (level 2 or 3), but the distributions suggest that they have different characteristics. We are currently conducting a case study on these errors for further language-specific improvements.

6 Conclusion and Future Work

In this study, a scheme was used to substantially simplify four abugidas, omitting most diacritics and merging the remaining characters. An SVM and an LSTM-based RNN were then used to recover the original texts, showing that the simplified abugidas could be recovered well. This illustrates the feasibility of encoding abugidas less redundantly, which could help with the development of more efficient input methods.

As for the future work, we are planning to include language-specific optimizations in the design of the simplification scheme and to improve the LSTM-based RNN by integrating dictionaries and increasing the amount of training data.

	Thai	Burmese	Khmer	Lao
$\text{Dev}_{\pm 3}$	96.1%	94.0%	94.6%	91.5%
$\text{Dev}_{\pm 5}$	**97.1%**	96.0%	**95.7%**	**93.1%**
$\text{Dev}_{\pm 7}$	97.0%	**96.3%**	**95.7%**	93.0%
Test	97.2%	96.1%	95.2%	93.1%
Leng.	76.0%	74.0%	77.6%	72.8%

Table 1: Top-1 recovery accuracy for the SVM. Here, "$\text{Dev}_{\pm m}$" represents the results for the development set when using N-gram ($N \in [1, m]$) features within m-grapheme windows of the simplified encodings, and "Test" represents the test set results when using the feature set that gave the best development set results. "Leng." shows the ratio of the number of characters in the simplified encodings compared with the original strings.

	Thai	Burmese	Khmer	Lao
SVM	97.2%	96.1%	95.2%	93.1%
$\text{RNN}^{@1}_{\oplus 4}$	97.2%	96.3%‡	95.5%‡	93.3%‡
$\text{RNN}^{@1}_{\oplus 8}$	97.3%†	96.4%‡	**95.6%**‡	**93.6%**‡
$\text{RNN}^{@1}_{\oplus 16}$	**97.4%**‡	**96.5%**‡	**95.6%**‡	**93.6%**‡
$\text{RNN}^{@2}_{\oplus 16}$	98.8%	98.4%	97.5%	96.6%
$\text{RNN}^{@4}_{\oplus 16}$	99.2%	98.8%	98.1%	97.7%
$\text{RNN}^{@8}_{\oplus 16}$	99.2%	98.9%	98.4%	97.9%

Table 2: Top-n accuracy on the test set for the LSTM-based RNN with an m-model ensemble ($\text{RNN}^{@n}_{\oplus m}$). Here, † and ‡ mean the RNN outperformed the SVM with statistical significance at $p < 10^{-2}$ and $p < 10^{-3}$ level, respectively, measured by bootstrap re-sampling.

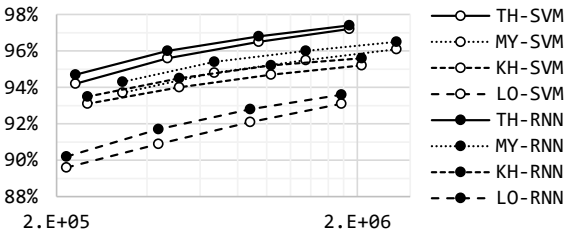

Figure 5: Top-1 accuracy on the test set (y-axis) for different training data sizes (x-axis, number of graphemes after simplification, logarithmic).

Level	0	1	2	3
Burmese	4.5%	51.0%	42.2%	2.2%
Khmer	22.5%	28.5%	16.3%	32.8%

Table 3: Recovery error distribution.

Acknowledgments

The experimental results of Burmese were evaluated by Dr. Win Pa Pa and Ms. Aye Myat Mon from *University of Computer Studies, Yangon*, Myanmar. The experimental resutls of Khmer were evaluated by Mr. Vichet Chea, Mr. Hour Kaing, Mr. Kamthong Ley, Mr. Vanna Chuon, and Mr. Saly Keo from *National Institute of Posts, Telecommunications and ICT*, Cambodia. We are grateful for their collaboration. We would like to thank Dr. Atsushi Fujita for his helpful comments.

References

Zheng Chen and Kai-Fu Lee. 2000. A new statistical approach to Chinese pinyin input. In *Proc. of ACL*. pages 241–247.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning* 20(3):273–297.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of machine learning research* 9(Aug):1871–1874.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Wei Jiang, Yi Guan, Xiaolong Wang, and Bingquan Liu. 2007. Pinyin-to-character conversion model based on support vector machines. *Journal of Chinese information processing* 21(2):100–105.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint*.

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

Lu Li, Xuan Wang, Xiaolong Wang, and Yanbing Yu. 2009. A conditional random fields approach to Chinese pinyin-to-character conversion. *Journal of Communication and Computer* 6(4):25–31.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. DyNet: The dynamic neural network toolkit. *arXiv preprint*.

Phavy Ouk, Ye Kyaw Thu, Mitsuji Matsumoto, and Yoshiyori Urano. 2008. The design of Khmer word-based predictive non-QWERTY soft keyboard for stylus-based devices. In *Proc. of VL/HCC*. pages 225–232.

Hammam Riza, Michael Purwoadi, Gunarso, Teduh Uliniansyah, Aw Ai Ti, Sharifah Mahani Aljunied, Luong Chi Mai, Vu Tat Thang, Nguyen Phuong Thai, Rapid Sun, Vichet Chea, Khin Mar Soe, Khin Thandar Nwet, Masao Utiyama, and Chenchen Ding. 2016. Introduction of the Asian language treebank. In *Proc. of O-COCOSDA*. pages 1–6.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of machine learning research* 15(1):1929–1958.

Hiroyuki Tokunaga, Daisuke Okanohara, and Shinsuke Mori. 2011. Discriminative method for Japanese kana-kanji input method. In *Proc. of WTIM*. pages 10–18.

Xuan Wang, Lu Li, Lin Yao, and Waqas Anwar. 2006. A maximum entropy approach to Chinese pinyin-to-character conversion. In *Proc. of SMC*. pages 2956–2959.

Ian H. Witten, Timothy C. Bell, Alistair Moffat, Craig G. Nevill-Manning, Tony C. Smith, and Harold Thimbleby. 1994. Semantic and generative models for lossy text compression. *The Computer Journal* 37(2):83–87.

Shaohua Yang, Hai Zhao, and Baoliang Lu. 2012. A machine translation approach for Chinese whole-sentence pinyin-to-character conversion. In *Proc. of PACLIC*. pages 333–342.