Simple Discriminative Training for Machine Transliteration

Canasai Kruengkrai, Thatsanee Charoenporn, Virach Sornlertlamvanich
National Electronics and Computer Technology Center
Thailand Science Park, Klong Luang, Pathumthani 12120, Thailand
{canasai.kruengkrai,thatsanee.charoenporn,virach.sornlertlamvanich}@nectec.or.th

Abstract

In this paper, we describe our system used in the NEWS 2011 machine transliteration shared task. Our system consists of two main components: simple strategies for generating training examples based on character alignment, and discriminative training based on the Margin Infused Relaxed Algorithm. We submitted results for 10 language pairs on standard runs. Our system achieves the best performance for English-to-Thai and English-to-Hebrew.

1 Introduction

We aim to develop a machine transliteration system that performs well in any given language pair without much effort in pre- and post-processing, and parameter tuning. To compare the performance of our system against state-of-the-art approaches, we participated in the machine transliteration shared task conducted as a part of the Named Entities Workshop (NEWS 2011), an IJCNLP 2011 workshop. Specifically, we focus on standard runs where only the corpus (containing parallel names) provided by the shared task is used for training. We submitted results for 10 language pairs.

2 Background

2.1 Motivation

As discussed in (Li et al., 2004), machine transliteration can be viewed as two levels of decoding: (1) segmenting the source language character string into transliteration units, and (2) relating the source language transliteration units with units in the target language by resolving different combinations of alignments and unit mappings. A transliteration unit could be one or more characters. Typically, the source and target language transliteration units are not given in the training corpus.

The process of machine transliteration is very similar to that of phrase-based statistical machine translation (SMT) (Koehn et al., 2003). As a result, a number of previous studies directly applied phrase-based SMT techniques to machine transliteration (Finch and Sumita, 2009; Rama and Gali, 2009; Finch and Sumita, 2010; Avinesh and Parikh, 2010). However, unlike word alignment in phrase-based SMT, character alignment in machine transliteration seems to be monotonic in which reordering of target language characters rarely occurs but is still possible in some language pairs.

After alignment, the target language transliteration units can be considered as tags (or labels) of the source language transliteration units. As a result, some previous studies viewed machine transliteration as simply as a sequence labeling problem (Aramaki and Abekawwa, 2009; Shishtla et al., 2009). With this problem setting, the system can apply any powerful discriminative training algorithm (e.g., Conditional Random Fields (CRFs) (Lafferty, 2001)) incorporated with rich features. Our system follows this research direction, but we pay more attention on how to extract appropriate transliteration units and train our model using the Margin Infused Relaxed Algorithm (MIRA) (Crammer et al., 2005; McDonald, 2006).

2.2 Problem Setting

Here, we formulate the process of machine transliteration based on discriminative learning. Given a character string $\boldsymbol{x}$ in the source language, we need to find the most likely character string $\hat{\boldsymbol{y}}$ out of all possible character strings in the target language. We express this process by:

$$\hat{\boldsymbol{y}} = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}} s(\boldsymbol{x}, \boldsymbol{y}; \mathbf{w}) , \qquad (1)$$

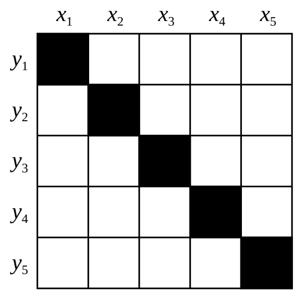

Figure 1: Ideal alignment.

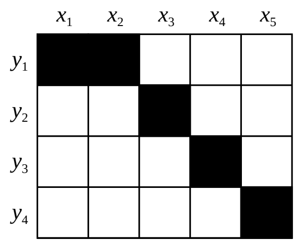

Figure 2: The source language character string ***x*** is longer than the target language character string ***y***. The aligner maps two source language characters to a single target language character.

where s denotes a discriminant function over a pair of a source language character string ***x*** and a hypothesized target language character string ***y*** given a parameter $\mathbf{w}$.

3 Strategies for Generating Training Examples

In this section, we describe how to generate training examples from a parallel name corpus. Our training example construction is based on character alignment.

At the first step, we can apply any word alignment tool commonly used in SMT. Given a training corpus containing parallel name pairs, we use the aligner to obtain initial character alignments. Figure 1 shows an ideal alignment example between the source language character string ***x*** and the target language character string ***y***. Now, assume that we have only one parallel name pair. Thus, our training example can be directly written as $(\langle x_1, y_1\rangle, \langle x_2, y_2\rangle, \ldots, \langle x_5, y_5\rangle)$.

Unfortunately, the lengths of parallel name pairs in the training corpus are typically unequal. The source language character string ***x*** could be shorter or longer than the target language character string ***y***. Figure 2 shows an example when ***x*** is longer than ***y***, and the aligner maps two source language characters to a single target language character, i.e., $\{x_1, x_2\} \rightarrow y_1$. To handle this case, we associate the position-of-character (POC) tags with the target language character. Our POC tags includes $\{\mathrm{B}, \mathrm{I}\}$, indicating the beginning and the intermediate positions, respectively. Our training example becomes $(\langle x_1, \mathrm{B}\text{-}y_1\rangle$, $\langle x_2, \mathrm{I}\text{-}y_1\rangle$, $\langle x_3, \mathrm{B}\text{-}y_2\rangle$, $\langle x_4, \mathrm{B}\text{-}y_3\rangle$, $\langle x_5, \mathrm{B}\text{-}y_4\rangle)$.

In practice, the aligner often yields incomplete alignments. Some target language characters could not be aligned to source language characters, and vice versa. To handle this case, we use simple heuristics by looking at neighboring alignments. We find unaligned characters in both the source and target character strings. If the previous alignment is already established, we expand it to the empty alignment. If the previous alignment is not available (e.g., the unaligned character occurs at the beginning position), we instead use the information from the next alignment.

Figure 3 shows an example when the aligner cannot map x_2 to any target language character. Based on our heuristics, we align x_2 to y_1. As a result, our training example is identical to that

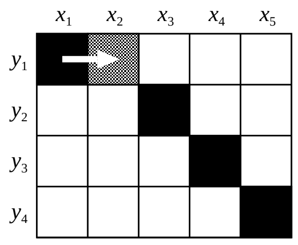

Figure 3: The aligner cannot map x_2 to any target language character. Based on the information from the previous alignment, we align x_2 to y_1.

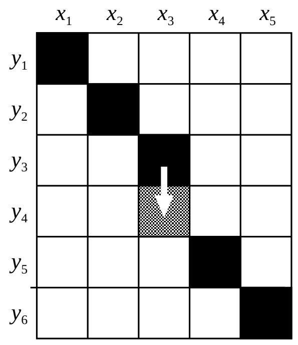

Figure 4: The aligner cannot map y_4 to any source language character. Based on the information from the previous alignment, we align y_4 to x_3.

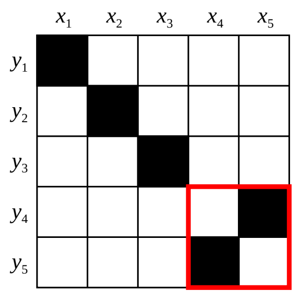

Figure 5: Reordering occurs in the target language characters. y_4 and y_5 are first merged into a single transliteration unit y_4y_5, and x_4 and x_5 are then aligned to B-y_4y_5 and I-y_4y_5, respectively.

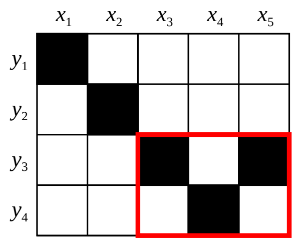

Figure 6: Another possible character reordering.

of Figure 2. Figure 4 shows another example when the aligner cannot map y_4 to any source language character. In this case, we align y_4 to x_3. Now, a single source language character is associated with two target language characters, i.e., $x_3 \rightarrow \{y_3, y_4\}$. As a result, we merge y_3 and y_4 into a single transliteration unit y_3y_4. Our training example becomes ($\langle x_1$, B-$y_1\rangle$, $\langle x_2$, B-$y_2\rangle$, $\langle x_3$, B-$y_3y_4\rangle$, $\langle x_4$, B-$y_5\rangle$, $\langle x_5$, B-$y_6\rangle$).

Note that character reordering can be found in the alignments. Figure 5 shows an example when reordering occurs in the target language characters. To be able to perform the monotone search in decoding, we merge y_4 and y_5 into a single transliteration unit y_4y_5. Our training example becomes ($\langle x_1$, B-$y_1\rangle$, $\langle x_2$, B-$y_2\rangle$, $\langle x_3$, B-$y_3\rangle$, $\langle x_4$, B-$y_4y_5\rangle$, $\langle x_5$, I-$y_4y_5\rangle$).

Figure 6 shows another possible character reordering. We use the same scheme as the previous example. Thus, our training example becomes ($\langle x_1$, B-$y_1\rangle$, $\langle x_2$, B-$y_2\rangle$, $\langle x_3$, B-$y_4y_5\rangle$, $\langle x_4$, I-$y_4y_5\rangle$, $\langle x_5$, I-$y_4y_5\rangle$). To summarize, we examine whether reordering occurs in the target language characters. If so, we merge those target language characters until the alignments become monotonic.

4 Learning and Decoding

The goal of our model is to learn a mapping from source language character strings $\boldsymbol{x} \in \mathcal{X}$ to target language character strings $\boldsymbol{y} \in \mathcal{Y}$ based on training examples of source-target language name pairs $\mathcal{D} = \{(\boldsymbol{x}_t, \boldsymbol{y}_t)\}_{t=1}^{T}$.

In our model, we apply a generalized version of MIRA (Crammer et al., 2005; McDonald, 2006) that can incorporate k-best decoding in the update procedure. From Equation (1), the linear discriminant function s becomes the dot product between a feature function $\mathbf{f}$ of the source language character string $\boldsymbol{x}$ and the target language character string $\boldsymbol{y}$ and a corresponding weight vector $\mathbf{w}$:

$$s(\boldsymbol{x}, \boldsymbol{y}; \mathbf{w}) = \langle \mathbf{w}, \mathbf{f}(\boldsymbol{x}, \boldsymbol{y}) \rangle \ . \tag{2}$$

In each iteration, MIRA updates the weight vector $\mathbf{w}$ by keeping the norm of the change in the weight vector as small as possible. With this framework, we can formulate the optimization problem as follows (McDonald, 2006):

$$\begin{aligned} &\mathbf{w}^{(i+1)} = \text{argmin}_{\mathbf{w}} \|\mathbf{w} - \mathbf{w}^{(i)}\| \\ &\text{s.t. } \forall \hat{\boldsymbol{y}} \in \text{best}_k(\boldsymbol{x}_t; \mathbf{w}^{(i)}) : \\ &s(\boldsymbol{x}_t, \boldsymbol{y}_t; \mathbf{w}) - s(\boldsymbol{x}_t, \hat{\boldsymbol{y}}; \mathbf{w}) \geq L(\boldsymbol{y}_t, \hat{\boldsymbol{y}}) \ , \end{aligned} \tag{3}$$

where $\text{best}_k(\boldsymbol{x}_t; \mathbf{w}^{(i)})$ represents a set of top k-best outputs given the weight vector $\mathbf{w}^{(i)}$. We generate $\text{best}_k(\boldsymbol{x}_t; \mathbf{w}^{(i)})$ using a dynamic programming search (Nagata, 1994). We measure $L(\boldsymbol{y}_t, \hat{\boldsymbol{y}})$ using the zero-one loss function. Our basic features operate over the window of ± 4 source language characters and the target language character bigrams.

5 Development and Final Results

In development, we were interested in how the quality of alignment affects the performance of transliteration because errors in alignment inevitably propagate to the learning phase. We used two popular alignment tools, including GIZA++[1] (Och and Ney, 2003) and BerkeleyAligner[2] (Liang et al., 2006). With their default parameter settings, GIZA++ yields better performance than BerkeleyAligner on all development data sets. As a result, our submitted primary runs on the test data sets are based on the resulting alignments from GIZA++. Our learning algorithm

[1]http://code.google.com/p/giza-pp
[2]http://code.google.com/p/berkeleyaligner

Language Pair	ACC	F-score	MRR	MAP$_{ref}$	Rank (# of all primary runs)
En→Ch	0.342	0.702	0.406	0.331	2 (7)
Ch→En	0.131	0.730	0.193	0.131	5 (6)
En→Th	**0.354**	**0.854**	**0.451**	**0.350**	**1** (2)
Th→En	0.284	0.841	0.402	0.283	2 (2)
En→Hi	0.436	0.870	0.538	0.435	3 (4)
En→Ta	0.432	0.896	0.553	0.430	2 (2)
En→Ka	0.398	0.878	0.502	0.397	2 (2)
En→Ba	0.455	0.887	0.557	0.453	2 (2)
En→Pe	0.643	0.943	0.744	0.629	2 (4)
En→He	**0.602**	**0.931**	**0.702**	**0.602**	**1** (2)

Table 1: Final results showing the "standard run" performance of our system on the test data sets. Language acronyms include En = English, Ch = Chinese, Th = Thai, Hi = Hindi, Ta = Tamil, Ka = Kannada, Ba = Bengali (Bangla), Pe = Persian, and He = Hebrew.

has two tunable parameters: the number of training iterations N and the number of top k-best outputs. We heuristically set $N = 10$ and $k = 5$ for all experiments.

Final results showing the "standard run" performance of our system on the test data sets are given in Table 1. Evaluation metrics include word accuracy in top-1 (ACC), fuzziness in top-1 (F-score), mean reciprocal rank (MRR), and MAP$_{ref}$ described in more detail in (Zhang et al., 2011). The table shows the scores of our primary runs, and the last column indicates our ranks in which we compare our scores with those of other participants.

Our system performs reasonably well across language pairs, except for Chinese-to-English back-transliteration. We achieve the best performance for English-to-Thai and English-to-Hebrew, and the second-best performance (in the cases that more than two primary runs were submitted) for English-to-Chinese and English-to-Persian.

References

Eiji Aramaki and Takeshi Abekawwa. 2009. Fast decoding and easy implementation: transliteration as sequential labeling. In *Proceedings of the 2009 Named Entities Workshop*, pages 65–68.

P. V. S. Avinesh and Ankur Parikh. 2010. Phrase-based transliteration system with simple heuristics. In *Proceedings of the 2010 Named Entities Workshop*, pages 81–84.

Koby Crammer, Ryan McDonald, and Fernando Pereira. 2005. Scalable large-margin online learning for structured classification. In *NIPS Workshop on Learning With Structured Outputs*.

Andrew Finch and Eiichiro Sumita. 2009. Transliteration by bidirectional statistical machine translation. In *Proceedings of the 2009 Named Entities Workshop*, pages 52–56.

Andrew Finch and Eiichiro Sumita. 2010. Transliteration using a phrase-based statistical machine translation system to re-score the output of a joint multigram model. In *Proceedings of the 2010 Named Entities Workshop*, pages 48–52.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT-NAACL*, pages 48–54.

John Lafferty. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*, pages 282–289.

Haizhou Li, Min Zhang, and Su Jian. 2004. A joint source-channel model for machine transliteration. In *Proceedings of ACL*, pages 159–166.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *Proceedings of HLT-NAACL*, pages 104–111.

Ryan McDonald. 2006. *Discriminative Training and Spanning Tree Algorithms for Dependency Parsing*. University of Pennsylvania, PhD Thesis.

Masaki Nagata. 1994. A stochastic japanese morphological analyzer using a forward-DP backward-A* n-best search algorithm. In *Proceedings of COLING*, pages 201–207.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Comput. Linguist.*, 29:19–51.

Taraka Rama and Karthik Gali. 2009. Modeling machine transliteration as a phrase based statistical machine translation problem. In *Proceedings of the 2009 Named Entities Workshop*, pages 124–127.

Praneeth Shishtla, V. Surya Ganesh, Sethuramalingam Subramaniam, and Vasudeva Varma. 2009. A language-independent transliteration schema using character aligned models at news 2009. In *Proceedings of the 2009 Named Entities Workshop*, pages 40–43.

Min Zhang, A Kumaran, and Haizhou Li. 2011. Whitepaper of news 2011 shared task on machine transliteration. `http://translit.i2r.a-star.edu.sg/news2011`.