論文 / 著書情報
Article / Book Information

Title	A New Hybrid Method for Machine Transliteration
Authors	Dong Yang, Paul Dixon, Sadaoki Furui
出典 / Citation	IEICE Transactions on Information and Systems, Vol. E93-D, No. 12, pp. 3377-3383
発行日 / Pub. date	2010, 12
URL	http://search.ieice.org/
権利情報 / Copyright	本著作物の著作権は電子情報通信学会に帰属します。 Copyright (c) 2010 Institute of Electronics, Information and Communication Engineers.

PAPER

A New Hybrid Method for Machine Transliteration

Dong YANG[†a)], **Paul DIXON**[†], *Nonmembers, and* **Sadaoki FURUI**[†], *Fellow*

SUMMARY This paper proposes a new hybrid method for machine transliteration. Our method is based on combining a newly proposed two-step conditional random field (CRF) method and the well-known joint source channel model (JSCM). The contributions of this paper are as follows: (1) A two-step CRF model for machine transliteration is proposed. The first CRF segments a character string of an input word into chunks and the second one converts each chunk into a character in the target language. (2) A joint optimization method of the two-step CRF model and a fast decoding algorithm are also proposed. Our experiments show that the joint optimization of the two-step CRF model works as well as or even better than the JSCM, and the fast decoding algorithm significantly decreases the decoding time. (3) A rapid development method based on a weighted finite state transducer (WFST) framework for the JSCM is proposed. (4) The combination of the proposed two-step CRF model and JSCM outperforms the state-of-the-art result in terms of top-1 accuracy.

key words: *machine transliteration, two-step CRF, joint optimization, system combination*

1. Introduction

There are more than 6000 languages in the world and 10 languages have more than 100 million native speakers. With the information revolution and globalization, systems that support multi-language processing and language translation become urgent demands. In multi-language systems, the out-of-vocabulary (OOV) problems caused by named entities and technical terms from different languages have become a serious issue. The translation of these terms between alphabetic and syllabic languages is usually performed through transliteration, which tries to preserve the pronunciation in the original language.

Most non-alphabetic languages have standard Romanization systems that map an item in the source language to its alphabetic format. So transliteration from a non-alphabetic language to an alphabetic language is not challenging. However, the other direction (an alphabetic language to a non-alphabetic language) is usually a difficult task, due to the mismatch of written units and a lack of standard conversion systems (functioning as reverse Romanization systems).

For example, in Chinese, foreign words are written with Chinese characters; in Japanese, foreign words are usually written with special syllabic characters (specially designed for imported words) called Katakana; examples are given in Fig. 1.

An intuitive transliteration method [1] is to firstly convert a source word into phonemes, then find the corresponding phonemes in the target language, and finally convert them to the target language's written system. There are two reasons why this method doesn't work well: first, the named entities are usually OOVs with diverse origins and this makes the grapheme-to-phoneme conversion very difficult; second, the transliteration is usually not only determined by the pronunciation, but also affected by the grapheme, e.g. the Japanese example in Fig. 1, a Katakana "ru" is inserted into the Japanese transliteration because of the existence of r in the grapheme, despite the fact that there is no /r/ in the English pronunciation.

Direct orthographical mapping (DOM), which performs the transliteration between two languages directly without using any intermediate phonemic mapping, has recently been gaining more attention in the transliteration research community, and it is also the "Standard Run" of the "NEWS 2009 Machine Transliteration Shared Task" [2]. In this paper, we try to make our system satisfy the standard evaluation condition, which requires that the system uses the provided parallel corpus (without pronunciation) only, and cannot use any other bilingual or monolingual resources.

The source channel model and joint source channel model (JSCMs) [3] have been proposed for DOM, and they try to model $P(T|S)$ and $P(T,S)$, respectively, where T and S denotes the words in the target and source languages. [4] modified the JSCM to incorporate different contextual information into the model for Indian languages. Several methods based on discriminative models have been proposed for machine transliteration [2], [5]–[7], and one of them [2], [6] has achieved the state-of-the-art result. One of the problems with these methods is that the algorithm is usually very complex and takes a very long time to implement.

For the "NEWS 2009 Machine Transliteration Shared Task" we have proposed a new two-step conditional random

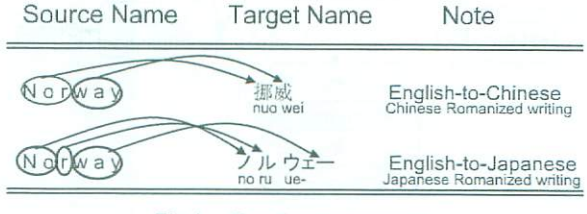


Fig. 1 Transliteration examples.

Manuscript received March 17, 2010.
Manuscript revised July 28, 2010.

[†]The authors are with the Dept. of Computer Science, Tokyo Institute of Technology, Tokyo, 152–8552 Japan.
a) E-mail: raymond@furui.cs.titech.ac.jp

DOI: 10.1587/transinf.E93.D.3377

field (CRF) model for transliteration [8], in which the first step is to segment a word in the source language into character chunks and the second step is to perform a context-dependent mapping from each chunk into one written unit in the target language. We summarize the method in this paper and report our recent progress on further improving the two-step CRF method. We propose jointly optimizing the two steps together and a fast decoding algorithm to speed up the joint search.

Following the two-step CRF model, we propose a method to rapidly implement the JSCM using weighted finite state transducers (WFSTs). In the implementation, we convert all the sub-tasks into WFSTs, and use publicly available toolkits to perform the standard operations, such as compositions and n-best search. Although the WFST has already been used for machine transliteration [1], it was implemented for the source channel model (not for the JSCM) and it could not deal with character chunks.

We also propose a method to combine the two-step CRF model and the JSCM.

The rest of this paper is organized as follows: Section 2 describes the system structure, and Sect. 3 introduces our proposed two-step CRF model, followed by Sect. 4 which describes our joint optimization method and its fast decoding algorithm; Section 5 introduces our rapid implementation of a JSCM system; System combination and experiments are described in Sects. 6 and 7, respectively; Section 8 discusses experimental results; the last section concludes this paper and points out future work. Although our method is language independent, we use an **English-to-Chinese** transliteration task in all the explanations and experiments.

2. System Structure

The structure of our system is shown in Fig. 2, details of which will be explained in the following sections. The basic procedure of our system is as follows. Parallel training data is aligned first, and then a WFST based JSCM system and a two-step CRF system are built from the aligned data, respectively. When performing the transliteration, outputs from the two systems are combined to obtain the final result.

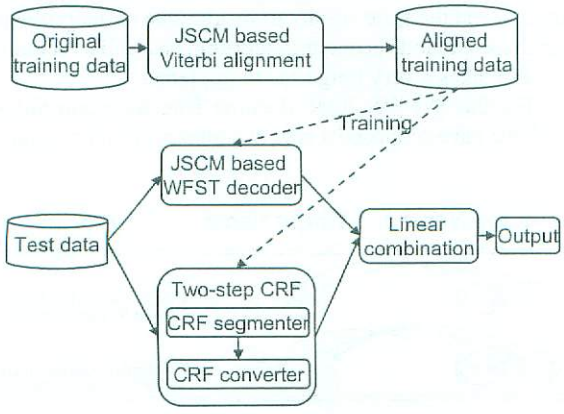


Fig. 2 System structure.

3. A Two-Step CRF Model for Machine Transliteration

We formalize machine transliteration as a two-step CRF tagging problem, as shown in Fig. 3. The first CRF segments an English word into chunks, and the second CRF converts each chunk to a Chinese character.

The alignment used for CRF training is obtained by the method described in Sect. 5.1.

3.1 CRF Introduction

A chain-CRF [9], which is an undirected graphical model, assigns a probability to a label sequence $L = l_1 l_2 \ldots l_T$, given an input sequence $C = c_1 c_2 \ldots c_T$,

$$P(L|C) = \frac{1}{Z(C)} exp\left(\sum_{t=2}^{T}\sum_{k}\lambda_k f_k(l_t, l_{t-1}, C, t)\right) \quad (1)$$

For the k^{th} feature, f_k denotes the feature function and λ_k is the parameter which controls the weighting. $Z(C)$ is a normalization term that ensures that the distribution sums to one. CRF training is usually performed by the L-BFGS algorithm [10] and decoding is performed by the Viterbi algorithm.

3.2 CRF Segmenter

In the CRF, the feature function describes a co-occurrence relation, and it is formally defined as $f_k(l_t, l_{t-1}, C, t)$ (Eq. (1)). f_k is usually a binary function, and takes the value 1 when both observation C and transition $l_{t-1} \rightarrow l_t$ are observed. In our segmentation tool, we use the following features:

- Uni-gram features: $C_{-2}, C_{-1}, C_0, C_1, C_2$
- Bi-gram features: $C_{-1}C_0, C_0C_1$

Here, C_0 is the current character, C_{-1} and C_1 denote the previous and next characters, and C_{-2} and C_2 are the characters located two positions to the left and right of C_0.

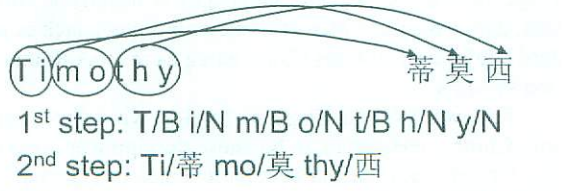


Fig. 3 A pictorial description of a CRF segmenter and converter example. The first line is a transliteration example, in which the mapping relation between an English word and a Chinese word is displayed. The second line is a segmentation example, and the tagging set is composed of two tags: B and N, in which B represents a starting position of a new chunk and N means a non-starting position. The third line is a converter example, in which the tagging set is composed of about 400 Chinese chracters, and each chunk is tagged as a character.

Fig. 4 An example of how to perform transliteration via one-step CRF, in which "-" represents a null output.

In our system participating in the "NEWS 2009 Machine Transliteration Shared Task", only top-1 segmentation is output to the following CRF converter.

3.3 CRF Converter

Similar to the CRF segmenter, the CRF converter has the format shown in Fig. 3.

In this CRF, the sequence to be tagged is composed of alphabet chunks, so the units here are chunks. For this CRF, we use the following features:

- Uni-gram features: CK_{-1}, CK_0, CK_1
- Bi-gram features: $CK_{-1}CK_0, CK_0CK_1$

where CK represents the English character chunk, and the subscript notation is the same as the CRF segmenter.

3.4 Two-Step CRF-Based Transliteration

It is possible to perform transliteration using a one-step CRF, which is shown in Fig. 4. The basic idea is to map an English character to a Chinese character directly, instead of an English chunk to a Chinese character. Some English characters correspond to Chinese characters, while others correspond to a null symbol "-".

Compared to the two-step CRF method, the one-step CRF method has two disadvantages. First, the training time is much longer. In the two-step CRF method, the CRF segmenter's computation is very short, and can be ignored compared to that of the CRF converter. If we compare the CRF converter and the one-step CRF, we find that they have an almost identical tag set (there is a null tag in the one-step CRF), but they have very different sequence length. In the one-step CRF, the basic unit is the English character, so the length is the number of English characters; in the CRF converter, the basic unit is the chunk, so the length corresponds to the number of chunks. The former lengths are 2.11 times of the latter on average. When training the CRF, the normalization factor is calculated as the sum over all possible tagging sequences. Its computation is affected greatly by the length of the observation sequence. Second, the two-step CRF covers longer dependencies than the one-step CRF. Comparing Fig. 3 and Fig. 4, as the features of two neighboring observation units, the former one based on chunks covers dependencies as long as 5 characters across, while the latter one based on characters only covers dependencies 2 characters across. The one-step CRF can use longer features to complement the problem, for example, 2 or 3 neighboring characters and their combinations can be added as features, and as a consequence the number of features is significantly increased.

Our preliminary experiments show that the one-step CRF method works substantially worse than the two-step CRF method, and its training time is ten times longer.

4. Joint Optimization and Its Fast Decoding Algorithm for the Two-Step CRF Method

4.1 Joint Optimization

We denote a word in the source language by S, a segmentation of S by A, and a word in the target language by T. Our goal is to find the best word $\hat{T}$ in the target language which maximizes the probability $P(T|S)$.

If we use only the best segmentation in the first CRF and the best output in the second CRF, the process is equivalent to

$$\begin{aligned} \hat{A} &= \arg\max_A P(A|S) \\ \hat{T} &= \arg\max_T P(T|S, \hat{A}), \end{aligned} \tag{2}$$

where $P(A|S)$ and $P(T|S, A)$ represent two CRFs, respectively. This method considers the segmentation and the conversion as two independent steps. A major limitation is that, if the segmentation from the first step is wrong, the error propagates to the second step, and the error is very difficult to recover from.

To alleviate the segmentation error problem, we propose a new method to jointly optimize the two-step CRF, which can be written as:

$$\begin{aligned} \hat{T} &= \arg\max_T P(T|S) \\ &= \arg\max_T \sum_A P(T, A|S) \\ &= \arg\max_T \sum_A P(A|S)P(T|A, S) \end{aligned} \tag{3}$$

The joint optimization considers multiple segmentation possibilities and sums the probabilities over alternative segmentations which generate the same output. Since it considers the segmentation and conversion in a unified framework, this method is robust against segmentation errors.

4.2 N-Best Approximation

In the process of finding the best output using Eq. (3), exact inference by listing all possible candidates and summing the probabilities over all possible segmentation is intractable because of the exponential computation complexity with the source word's increasing length.

In the segmentation step, the number of possible segmentations is 2^N, where N is the length of the source word and 2 is the size of the tagging set. In the conversion step, the number of possible candidates is $M^{N'}$, where N' is the

number of chunks from the 1st step and M is the size of the tagging set. M is usually large, e.g., about 400 in Chinese and 50 in Japanese, and it is impossible to list all the candidates.

Our analysis shows that beyond the 10th segmentation candidate, almost all the probabilities of the candidates in both steps fall below 0.01. Therefore we decided to generate top-10 results for both steps to approximate Eq. (3).

4.3 Fast Decoding Algorithm

As introduced in the previous subsection, in the whole decoding process we need to perform n-best CRF decoding in the segmentation step and 10 n-best CRF decoding in the second CRF. Is it really necessary to perform the second CRF decoding for all the segmentations? The answer is "No" for candidates with low probabilities. Hence we propose a no-loss fast decoding algorithm that determines when to stop performing the second CRF decoding.

Suppose we have a list of segmentation candidates which are generated by the 1st CRF, ranked by probabilities $P(A|S)$ in descending order $A : A_1, A_2, \ldots, A_N$ and we are performing the 2nd CRF decoding starting from A_1. Up to A_k, we get a list of candidates $T : T_1, T_2, \ldots, T_L$, ranked by probabilities in descending order. If we can guarantee that, even performing the 2nd CRF decoding for all the remaining segmentations $A_{k+1}, A_{k+2}, \ldots, A_N$, the top 1 candidate does not change, then we can stop decoding.

We can show that the following formula is the stop condition:

$$P(T_1|S) - P(T_2|S) > 1 - \sum_{j=1}^{k} P(A_j|S). \tag{4}$$

This means that the probability of all the remaining segmentation candidates is smaller than the probability difference between the best and the second best candidates; that is, even if all the remaining probabilities are added to the second best candidate, it still cannot overturn the top candidate.

The stop condition here has no approximation nor predefined assumption and it is a no-loss fast decoding algorithm. Along with the top-1 candidate, we can get a list of n-best results.

5. Rapid Development of a JSCM System

The JSCM describes how the source words and target words are generated simultaneously [3]:

$$\begin{aligned} P(S,T) &= P(s_1, s_2, \ldots, s_k, t_1, t_2, \ldots, t_k) \\ &= P(< s,t >_1, < s,t >_2, \ldots, < s,t >_k) \\ &= \prod_{k=1}^{K} P(< s,t >_k \mid < s,t >_1^{k-1}) \end{aligned} \tag{5}$$

where $S = (s_1, s_2, \ldots, s_k)$ is a word in the source langauge, $T = (t_1, t_2, \ldots, t_k)$ is a word in the target language, and s_k, t_k are either a written unit or several written units. An example is given in Fig. 5. From Eq. (5), we can see that $P(S,T)$ can be calculated in the same way as a language model, which can be approximated by an n-gram language model.

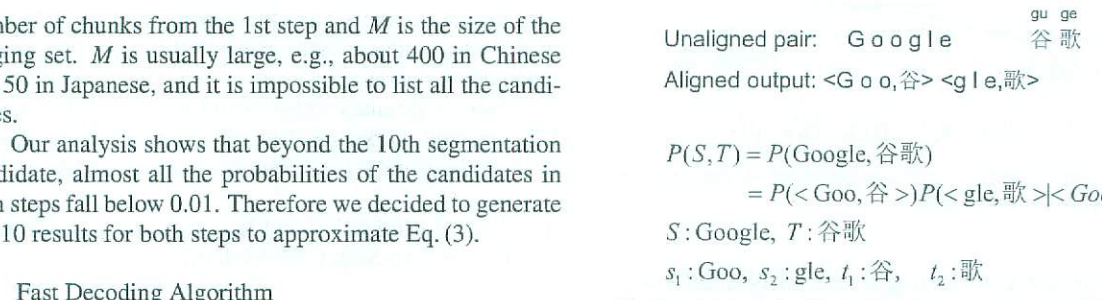


Fig. 5 An example of the aligned output, and how the JSCM probability is calculated.

5.1 Alignment

Given a parallel corpus composed of English and Chinese word pairs, the first step is to obtain the alignment which represents mapping relations between the two languages. Since there are no order changes in the alignment, what we have to do is to assign an English word and a Chinese word with the same number of units. We can either insert null characters into the Chinese word, or group English characters into chunks. In this paper, we perform the alignment using the latter method, grouping English characters into chunks.

In the alignment procedure, the initial alignments are created by performing random alignments across the training set, and then n-gram statistics are obtained. These statistics are used in Viterbi alignment to generate a new segmentation. Our method is almost the same as [3] which can be summarized as follows:

1. Initialize a random alignment
2. Calculate an n-gram model
3. E-step: apply the n-gram model to realign each entry in the corpus
4. M-step: update n-gram probabilities
5. Repeat 3 and 4 until the alignment converges

An example after the alignment is given in Fig. 5, in which the method for calculating a joint probability is also shown.

5.2 WFST Decoder Construction

The decoding problem in the JSCM can be written as:

$$\hat{T} = \arg\max_{T} P(S,T). \tag{6}$$

We implement the JSCM system based on the WFST framework. WFSTs are widely used in the areas of speech and language processing. Construction of the WFST and

searching the WFST are both performed using standard operations, such as composition and shortest path in the OpenFst toolkit [11]. The advantages of a WFST approach are rapid system design and implementation. Because there is no need to write a specific decoder, our development focuses on how to represent problems by WFSTs .

After generating the alignments, a *corpus* to train the transliteration WFST is built. Each aligned word pair is converted to a sequence of transliteration alignment pairs $\langle s,t\rangle_1, \langle s,t\rangle_2, \ldots \langle s,t\rangle_k$, where each s can be a chunk of one or more English characters and t is assumed to be a single Chinese character.

The entire set of alignments is used to train an n-gram language model, treating each of the pairs as a *word*. In the evaluation we used the MITLM toolkit [12] to build an n-gram model with modified Kneser-Ney smoothing. The language model is converted to a standard weighted acceptor representation. The acceptor is then converted to a transducer by simply splitting the label into s_k and t_k components and setting them respectively as input and output to give a transducer M [13].

Since the inputs for M are English character chunks, the second WFST T is constructed to convert characters into chunks to allow transliteration from a sequence of individual characters. T starts with a single state and for each English chunk s a path is added to map the string of individual character input to chunks on the output. These characters are also chunks themselves and an arc is added for each character that performs a simple one-to-one mapping, as shown in Fig. 6.

An input word is converted to a linear chain acceptor I, which contains an arc for each character in the input word. The three transducers are composed together and the best transliteration can be found using:

$$bestpath\,(\pi(I \circ T \circ M)) \tag{7}$$

Where $\circ$ denotes the composition operator and π is the projection which converts a transducer to an acceptor with the output labels, and further compacting can be achieved by applying epsilon removal. The system can be extended to generate n–best paths by using the n-shortest paths algorithm with determinization to ensure that top n results contain different transliteration candidates.

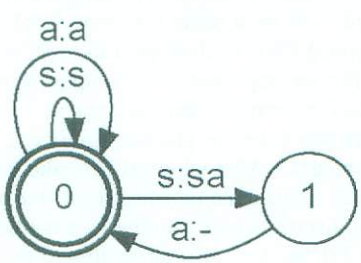


Fig. 6 An example of the WFST T, in which arcs for a chunk "sa" are constructed.

6. System Combination

We propose a method to combine the two-step CRF model and the JSCM. From the two-step CRF model we get a conditional probability $P_{CRF}(T|S)$ and from the JSCM we get a joint probability $P(S,T)$, and a conditional probability of $P_{JSCM}(T|S)$ can be calculated as follows:

$$P_{JSCM}(T|S) = \frac{P(T,S)}{P(S)} = \frac{P(T,S)}{\sum_T P(T,S)}. \tag{8}$$

They are used in our combination method as:

$$P(T|S) = \lambda P_{CRF}(T|S) + (1-\lambda) P_{JSCM}(T|S) \tag{9}$$

where λ denotes the interpolation weight which is set by using development data.

7. Experiments

7.1 Evaluation Metric

We use the first three metrics from [2] to measure the performance of our system. They are roughly described below; details can be found in [2].

1. Top-1 ACC: word accuracy of the top-1 candidate

This measures the correctness of the first generated transliteration. Sometimes there are several correct references for a word. ACC = 1 means all top candidates are correct transliterations i.e. they match one of the references, and ACC = 0 means that none of the top candidates are correct.

2. Mean F-score: fuzziness in the top-1 candidate, how close the top-1 candidate is to the closest reference

This measures how different, on average, the top transliteration candidate is from its closest reference. Precision and Recall are calculated based on the length of the longest common subsequence (LCS) between a candidate and a reference.

3. MRR: mean reciprocal rank, 1/MRR tells approximately the average rank of the correct result

Since our decoding algorithm aims at obtaining the best result in the first place, the Top-1 ACC is used as our major metric.

7.2 Corpus Used in Experiments

We use the training, development and test sets of NEWS 2009 data for English-to-Chinese in our experiments as detailed in Table 1. It is a parallel corpus and is not aligned.

Table 1 Corpus size (number of word pairs).

Training data	Development data	Test data
31961	2896	2896

7.3 Comparison with the JSCM

We first compared our two-step CRF method,which uses only the best candidates in both CRF steps as shown in Eq. (2), with the well known JSCM. We then added the joint optimization as shown in Eq. (3) to the two-step CRF method and compared it with the JSCM. As shown in Table 2, the proposed joint decoding method improved the top-1 ACC from 0.670 to 0.708, and it worked as well as or even better than the well known JSCM in all three measurements.

Our experiments also show that the decoding time can be reduced significantly using our fast decoding algorithm. As we have explained, without fast decoding, we need 11 CRF n-best decoding for each word. The number can be reduced to 3.53 (1 from "the first CRF" +2.53 from "the second CRF") via the fast decoding algorithm.

7.4 System Combinations

We use the combination methods described in Sect. 6. As shown in Table 3, the linear combination of two sytems further improves the top-1 ACC to 0.720, and it outperforms the best reported "Standard Run" [2], [6] result 0.717 (The reported best "Standard Run" achieved 0.731 but it used target language phoneme information which required a monolingual dictionary; as a result it was not a standard run).

7.5 Statistical Significance Tests

For the top-1 accuracy results in Tables 2 and 3, we have performed the binomial statistical significance tests. We define 1% as the strong significance level, and 5% as the weak significance level. The significance test results are shown in Table 4. In most of the tests, the null hypothesis is rejected.

Table 2 Comparison of the joint optimization with the baseline method and the JSCM.

Measure	Top-1 ACC	Mean F-score	MRR
CRF (1-best)	0.670	0.869	0.750
CRF (Joint optimization)	**0.708**	**0.885**	**0.789**
JSCM	0.706	0.882	**0.789**

Table 3 Model combination results.

Measure	Top-1 ACC	Mean F-score	MRR
CRF (1-best) + JSCM	0.713	0.883	0.794
CRF (Joint optim--ization) + JSCM	**0.720**	0.888	**0.797**
state-of-the-art [2], [6]	0.717	**0.890**	0.785

Table 4 Statistical significance test results: whether System II is better than System I (strong significance ○, weak significance △, and otherwise ×). Note that since we do not have the top-1 transliteration output of the state-of-the-art method [2], [6], we cannot perform significance test with it.

System II / System I	JSCM	CRF-J	1+2	2+3
1 CRF (1-best)	○	○	○	○
2 JSCM		×	○	○
3 CRF-J (Joint)			○	○
1 + 2				△

8. Discussions

Since the JSCM was proposed in machine transliteration [3], it has become the most well known statistical method because of its simplicity and excellent performance. The reason behind its effectiveness is that there are very good language model smoothing algorithms to adjust the conditional probabilities used in Eq. (5).

The performance of the proposed two-step CRF method is as good as or even better than JSCM. The main difference between our CRF method and the JSCM exists in the information used. The CRF uses the immediate previous and next chunks, while the JSCM tri-gram implementation uses previous two chunks. Although the CRF uses shorter histories than the JSCM, it uses future chunks. It is possible for CRF to use longer histories, but computation may then become a problem. Our CRF training took 13 hours on an 8-core computer, and including longer histories should require a significantly longer training time.

Combining the CRF and JSCM leads to better performance, probably because the feature sets that they use complement each other. As a result, the combination leads to the best performance, as shown in our experiment.

In the decoding part, it is also possible to use a thresholding method to further reduce the computational cost. We have tried the following method. Since the JSCM is much simpler, we use it as our main system; when $P_{JSCM}(T_{top-1}|S) > threshold$, we do not use the 2-step CRF system, and when $P_{JSCM}(T_{top-1}|S) <= threshold$, we combine the JSCM and the 2-step CRF. By setting the threshold to an appropriate value, we have found that the 2-step CRF decoding needs to be performed for only about 35.1% of the test data, and we still obtain the same result.

9. Conclusions and Future Work

In this paper, we have presented a new two-step CRF method for machine transliteration, along with its new joint optimization method and a fast decoding algorithm. We have also presented a WFST-based method to rapidly develop a general JSCM model. The proposed CRF method works as well as or even better than the JSCM. By combining the proposed CRF method with the JSCM, the performance was further improved and it outperformed the state-of-the-art result in terms of top-1 accuracy.

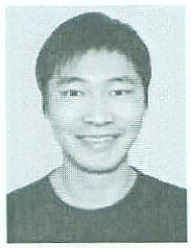
In future work we are planning to extend our method to other languages. Also we want to make use of acoustic information in machine transliteration. We are currently investigating discriminative training as a method to further improve the JSCM. It is also interesting to use some other discriminative models instead of the CRF model, such as LDCRF.

Acknowledgements

The corpus used in the experiments of this paper is from [3].

References

[1] K. Knight and J. Graehl, "Machine transliteration," Computational Linguistics, vol.24, pp.599–612, 1998.
[2] H. Li, A. Kumaran, V. Pervouchine, and M. Zhang, "Report of news 2009 machine transliteration shared task," Proc. 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009), Suntec, Singapore, pp.1–18, Association for Computational Linguistics, Aug. 2009.
[3] H. Li, M. Zhang, and J. Su, "A joint source-channel model for machine transliteration," Proc. 42nd Annual Meeting on Association for Computational Linguistics, pp.159–166, July 2004.
[4] A. Ekbal, S.K. Naskar, and S. Bandyopadhyay, "A modified joint source-channel model for transliteration," Proc. COLING/ACL, pp.191–198, 2006.
[5] K. Bellare, K. Crammer, and D. Freitag, "Loss-sensitive discriminative training of machine transliteration models," Proc. Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Student Research Workshop and Doctoral Consortium, pp.61–65, Association for Computational Linguistics, Boulder, Colorado, June 2009.
[6] S. Jiampojamarn, A. Bhargava, Q. Dou, K. Dwyer, and G. Kondrak, "Directl: A language independent approach to transliteration," Proc. 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009), pp.28–31, Association for Computational Linguistics, Suntec, Singapore, Aug. 2009.
[7] D. Zelenko, "Combining mdl transliteration training with discriminative modeling," Proc. 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009), pp.116–119, Association for Computational Linguistics, Suntec, Singapore, Aug. 2009.
[8] D. Yang, P. Dixon, Y.C. Pan, T. Oonishi, M. Nakamura, and S. Furui, "Combining a two-step conditional random field model and a joint source channel model for machine transliteration," Proc. 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009), pp.72–75, Association for Computational Linguistics, Suntec, Singapore, Aug. 2009.
[9] J. Lafferty, A. McCallum, and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," Proc. International Conference on Machine Learning, pp.282–289, 2001.
[10] H.M. Wallach, Efficient training of conditional random fields, M. Thesis, University of Edinburgh, 2002.
[11] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri, "*OpenFst*: A general and efficient weighted finite-state transducer library," Proc. Ninth International Conference on Implementation and Application of Automata, pp.11–23, 2007.
[12] B.J. Hsu and J. Glass, "Iterative language model estimation: Efficient data structure," Proc. Interspeech, pp.841–844, 2008.
[13] D. Caseiro, I. Trancosoo, L. Oliveira, and C. Viana, "Grapheme-to-phone using finite state transducers," Proc. 2002 IEEE Workshop on Speech Synthesis, pp.841–844, 2002.



Dong Yang is a Ph.D. candidate in the Department of Computer Science, Tokyo Institute of Technology, Japan. He received the B.E. degree in Electronic Engineering from Tsinghua University in 2005 and M.E. degree in Computer Science from Tokyo Institute of Technology in 2007, respectively. His current research interests include speech recognition and natural language processing.

Paul Dixon received Ph.D. and BEng degrees from the University of Birmingham (UK) in 2007 and 2000 respectively. Since 2006 he has been a researcher at Tokyo Institute of Technology conducting research and development of a WFST based speech recognition engine.

Sadaoki Furui received B.S., M.S. and Ph.D. degrees in mathematical engineering and instrumentation physics from Tokyo University, Tokyo, Japan in 1968, 1970 and 1978, respectively. He is engaged in a wide range of research on speech analysis, speech recognition, speaker recognition, speech synthesis, and multimodal human-computer interaction and has authored or coauthored over 800 published articles. He has received Paper Awards and Achievement Awards from the IEEE, the IEICE, the ASJ, the ISCA, the Minister of Science and Technology, and the Minister of Education, and the Purple Ribbon Medal from Japanese Emperor.