Direct Combination of Spelling and Pronunciation Information for Robust Back-Transliteration

Slaven Bilac and Hozumi Tanaka

Tokyo Institute of Technology
Ookayama 2-12-1, Meguro, 152-8552 Tokyo, Japan
{sbilac,tanaka}@cl.cs.titech.ac.jp,

Abstract. Transliterating words and names from one language to another is a frequent and highly productive phenomenon. For example, English word *cache* is transliterated in Japanese as キャッシュ "`kyasshu`". Transliteration is information losing since important distinctions are not always preserved in the process. Hence, automatically converting transliterated words back into their original form is a real challenge. Nonetheless, due to its wide applicability in MT and CLIR, it is an interesting problem from a practical point of view.
In this paper, we demonstrate that back-transliteration accuracy can be improved by directly combining grapheme-based (i.e. spelling) and phoneme-based (i.e. pronunciation) information. Rather than producing back-transliterations based on grapheme and phoneme model independently and then interpolating the results, we propose a method of first combining the sets of allowed rewrites (i.e. edits) and then calculating the back-transliterations using the combined set. Evaluation on both Japanese and Chinese transliterations shows that direct combination increases robustness and positively affects back-transliteration accuracy.

1 Introduction

With the advent of technology and increased flow of goods and services, it has become quite common to integrate new words from one language to another. Whenever a word is adopted into a new language, pronunciation is adjusted to suit the phonetic inventory of the language. Furthermore, the orthographic form of the word is modified to allow representation in the target language script. For example, English word *cache* is transliterated in Japanese as キャッシュ "`kyasshu`".[1] In similar fashion, a proper noun Duncan is transliterated as 桔刃 "`deng4ken3`" in Chinese.[2] This process of acquisition and assimilation of a new word into an existing writing system is referred to as transliteration [1].

[1] We use *italics* to transcribe the English words, while Japanese transliterations (e.g. キャッシュ) are given with romaji (i.e. roman alphabet) in "`typewriter`" font (e.g. "`kyasshu`"). The romanization used follows [1], thus closely reflecting English–like pronunciation with long vowels transcribed as "`aa`" rather than "`ā`".

[2] Chinese transliterations are given with the PinYin romanization with numerical tone codes.

Since integration of new words is a very productive process, it often happens that they are not recorded in machine or human dictionaries. Therefore, it is impossible to rely on dictionary lookup to find the transliteration pairs. Inability to find a target language equivalent represents a major problem in Machine Translation (MT) since it can cause translation failures. Furthermore, transliteration represents a serious problem in the area of Cross-Language Information Retrieval (CLIR) where new technical terms are frequently used in the queries and thus greatly affect performance [2, 3].

Back-transliteration is the transliteration back into the original language. It is generally more difficult than transliteration. Increase in difficulty results from the fact that various distinctions, present in the source language, are not always preserved when the word is transliterated into the target language. For example, Japanese has only five basic vowels and no /θ/ or /ð/[3] sounds. Non-existent sounds are replaced with their closest equivalents. Consequently, the following three English words: *bass*,*bath* and *bus* are transliterated as バス "`basu`".[4] A system trying to back-transliterate バス has therefore three valid choices which cannot be disambiguated in the absence of additional contextual information.

Transliterated words are normally written in katakana, one of three major character types making up the Japanese writing system. While other vocabulary (i.e. animal names or onomatopoeic expressions) can also be written in katakana, the fact that something is written in katakana is a good hint that it might be a transliterated foreign word or a name. Thus, unlike Arabic, Chinese or Korean, where a big part of the back-transliteration problem is identifying candidate transliterations [4–6], in Japanese back-transliteration can be directly applied to any katakana strings absent from the bilingual dictionary. In the evaluation on the Chinese data set, we avoid the problem of identifying the transliteration candidates since we train and evaluate on already extracted transliteration pairs. However, a real back-transliteration application would have to address this issue.

Previously, [7] proposed a hybrid back-transliteration method combining grapheme-based (i.e. spelling) and phoneme-based (i.e. pronunciation) information and demonstrated significant improvements over methods relying only on a single information source. In this paper, we show that the back-transliteration accuracy can be further improved by altering the manner in which grapheme-based and phoneme-based information is combined in the transliteration process and show that our method can easily be applied to Chinese (and, implicitly, other languages). Rather than producing back-transliterations based on grapheme and phoneme model independently and then interpolating the results, we propose a method of first combining the sets of allowed rewrites (i.e. edits) and then calculating back-transliterations using the combined set. Evaluation on both Japanese and Chinese transliterations shows that the manner in which grapheme-based and phoneme-based information are combined can significantly affect the system performance.

[3] All phonemes given in // are written in IPA symbols.

[4] Here /θ/ is replaced with /s/, and /æ/ is replaced with /a/.

The reminder of this paper is organized as follows: in Sect. 2 we review previous research. Sect. 3 outlines the proposed transliteration model and the system implementation. Finally, Sect. 4 gives an evaluation and a discussion of the results obtained, whereas Sect. 5 gives the conclusion.

2 Previous Research

Transliteration has received significant attention from researchers in recent years. Commonly, the source language considered is English and the target language is an Asian language void of an alphabetic writing system (e.g. Japanese, Korean, Chinese). Based on the underlying model, previous approaches to (back-) transliteration can be roughly divided into grapheme- and phoneme-based.

In the grapheme-based (or direct) modeling framework, the English string is not converted into a phonemic representation before its alignment with the transliterated string. Several different methods have been proposed within this framework: a decision tree based transliteration model [8], a maximum entropy based model [9], etc. Recently, [10] proposed a joint source-channel model that simultaneously models both source and channel context. In this model, after initial alignment, the preceding context (bigram, trigram) of both English and Chinese aligned units is considered simultaneously. Its main advantage is that it can be used for both transliteration and back-transliteration.

For back-transliteration of Japanese, [11] propose a noisy channel model allowing for non-atomics edits [12]. The input string is broken down into arbitrary substrings, each of which is output independently (and possibly incorrectly). The best back-transliteration is chosen using a modified edit distance algorithm [13, 14]. Since the best transliteration is determined through comparison with dictionary entries, this method does not handle phrases directly. This is a significant shortcoming since a large percent of transliterated strings are phrases.

In the phoneme-based (or pivot) modeling approach the pronunciation, rather than the spelling of the original string, is considered as a basis for transliteration. Among several approaches proposed are: an HMM based transliteration model [5], a rule based model [15] and a machine-learned phonetic similarity model [3].

For Japanese, [1] employ a compositional noisy-channel model combining romaji-to-phoneme, phoneme-to-English and English word probability models. The combined structure is treated as a graph, and the top ranking strings are found using the *k-best* path algorithm [16]. However, in this model there is no consideration of context information although context is crucial in determining the correct pronunciation of a given phoneme.

Recently, [7] proposed a back-transliteration model combining statistical string segmentation with a hybrid grapheme-phoneme transliteration model. The segmentation of the input string decreases the calculation complexity and allows for correct back-transliteration even of strings containing abbreviations. Furthermore, by taking both the spelling and pronunciation of the original into account when modeling transliteration, the system is able to achieve higher accuracy. However, the manner of combination is suboptimal since back-transliterations are produced by each model independently and then the results are interpolated to obtain the final back-transliterations. In this paper we propose a different method of combining the grapheme- and phoneme-based models. The unit alignment sets (collections of allowed edits in the noisy channel model) are combined into one set and then, back-transliterations are produced based on this set. Since alignment sets obtained by each base model are different, their combination results in a more comprehensive alignment set that can successfully handle a larger number of transliterations.

3 Transliteration Model

As mentioned above, in Japanese transliterated words are normally written in katakana. However, we implement katakana to romaji conversion as a preprocessing module and view back-transliteration as a process starting with a romanized string.[5]

Given some string in romaji J_a, the goal is to find the English word (phrase) E_a that maximizes the probability $P(E_a|J_a)$. Applying the Bayes' rule and dropping the constant denominator we get $P(J_a|E_a) \times P(E_a)$, where $P(E_a)$ is the source model and $P(J_a|E_a)$ is the noisy channel. For the source model, we use a word-based model which can output words from a list of valid tokens E_a with a certain probability distribution $P(E_a)$. For the channel model we train several different models (Sec. 3.1), and then invert each to enable handling of the romaji input. The source model and each inverted channel model are then combined to obtain the back-transliterations.

3.1 Base Models

Grapheme-based model. In the grapheme-based model (GM) the English word is directly rewritten as a Japanese romaji string with the probability $P_g(J_a|E_a)$. Rewriting can be viewed as a sequential process where the first stage is the partitioning of the input string into sub-word units which are then rewritten according to some mapping rule (1). Rather than trying to estimate the probability of breaking up the string in a certain way, we allow all valid segmentations given the segments in our learned mapping set with equal probability. Thus, the resulting equation is simplified as (2). Under the assumption of segment independence, (2) can be further simplified so the resulting probability of outputting J_a can be rewritten as in (3).

$$P_g(J_a|E_a) = P(E_{a_1}, E_{a_2} \ldots E_{a_n}|E_a) \times \\ P_g(J_{a_1}, J_{a_2} \ldots J_{a_n}|E_{a_1}, E_{a_2} \ldots E_{a_n}) \ . \quad (1)$$

[5] Katakana is a syllabary, and each character corresponds to one or more alphabet letters (e.g. ア "a", マ "ma" or シ "shi", etc.). By romanizing the input, we allow the model to capture the similarities on the letter level (e.g. that m in English often maps to "m" in Japanese) and reduce the data sparseness problem.

$$P_g(J_a|E_a) \cong P_g(J_{a_1}, J_{a_2} \ldots J_{a_n}|E_{a_1}, E_{a_2} \ldots E_{a_n}) \ . \tag{2}$$

$$P_g(J_a|E_a) \cong \prod_{i=1}^{n} P_g(J_{a_i}|E_{a_i}) \ . \tag{3}$$

In Sect. 3.2 we describe how we get the set of allowed edits $(E_{a_i} \rightarrow J_{a_i})$ and how we go about assigning the probability to each $P_g(J_{a_i}|E_{a_i})$.

Phoneme-based model. In this model the channel is broken up into two stages: a) conversion of the English alphabet into English phonemes with some probability $P(E_p|E_a)$ and b) conversion of English phonemes into romaji with some probability $P(J_a|E_p)$. Hence, $P_p(J_a|E_a)$ can be rewritten as Eq. (4). Rather than manipulating these two distributions separately, we compute their composition to obtain a unique probability distribution $P_p(J_{a_i}|E_{a_i})$. The composition produces a set of edit pairs $(E_{a_i} \rightarrow J_{a_i})$ such that $(E_{a_i} \rightarrow E_{p_i})$ is a member of the first set and $(E_{p_i} \rightarrow J_{a_i})$ is a member of the second set for some E_{p_i} [19].[6]

$$P_p(J_a|E_a) \cong \prod_{i=1}^{n} P(J_{a_i}|E_{p_i}) \times \prod_{i=1}^{n} P(E_{p_i}|E_{a_i}) \ . \tag{4}$$

Since obtained mapping set does not contain any English phoneme units, all English alphabet strings can be rewritten directly into romaji without requiring their conversion into intermediate phoneme representation. This removes the requirement of having a pronunciation dictionary for the back-transliteration.[7] Note also that both GM and PM are dealing with the same types of edits (i.e. English alphabet to romaji $(E_{a_i} \rightarrow J_{a_i})$) and can be directly combined.

3.2 Training

The training consists of learning possible edits and their probabilities. We train the GM and PM sets independently and then combine them.

For the grapheme-based model, romanized Japanese strings are aligned with English strings directly using the non-weighted Levenshtein distance [13, 14]. After initial alignment, each atomic edit (i.e. one aligning unit pair) is expanded through combination with adjacent edits. By doing so, we add contextual information to each edit, which helps reduce the ambiguity associated with atomic edits [17]. For example, for the pair (*vinyl*, `biniru`) we get the following alignment:

$$v \rightarrow \texttt{b} \quad i \rightarrow \texttt{i} \quad n \rightarrow \texttt{n} \quad y \rightarrow \texttt{i} \quad l \rightarrow \texttt{r} \quad \_ \rightarrow \texttt{u}$$

[6] This is a slight simplification since E_{p_i} can actually be a concatenation of several $E_{p_{n..m}}$ with respective mappings $E_{p_{n..m}} \rightarrow J_{a_{m..n}}$.

[7] However, the pronunciation dictionary is still necessary for the training since the mapping set is obtained via intermediate use of English phoneme representations.

For $N = 1$, the following edits are then also added to the set:

$$vi \to \texttt{bi}, in \to \texttt{in}, vin \to \texttt{bin}, ny \to \texttt{ni}, iny \to \texttt{ini},$$
$$yl \to \texttt{ir}, nyl \to \texttt{nir}, yl \to \texttt{iru}, l \to \texttt{ru}$$

We collect a complete set of edits $\alpha_g \to \beta_g$ in the training set and assign the probability to each according to (5). Throughout, we distinguish edits that appear at the beginning or the end of the word or neither.

$$P(\alpha \to \beta) = \frac{count(\alpha \to \beta)}{count(\alpha)} \quad . \tag{5}$$

For the PM, we obtain the optimal romaji to English phoneme alignment using the Estimation Maximization (EM) algorithm [18]. After EM selects the optimal alignment, we proceed to expand the set of individual alignments with N adjacent units as above to obtain a set of possible rewrites $\alpha_{e_p} \to \beta_{j_a}$. This process is repeated to obtain the set of all possible mappings of English alphabet strings into phoneme strings $\alpha_{e_a} \to \beta_{e_p}$. Each input α_{e_a} with all its mappings β_{e_p} and corresponding probabilities $P(\beta_{e_p}|\alpha_{e_a})$ is converted into a Weighted Finite State Transducer (WFST) T_p with α_{e_a} as inputs, β_{e_p} as outputs [19, 1] and transition costs as negative logs of probabilities. WFST T_p is then composed with a WFST T_c encoding the complete set of mappings $\alpha_{e_p} \to \beta_{j_a}$ to obtain the set of all possible rewrites of English alphabet strings α_p into romaji strings β_p based on the PM.

Directly combined grapheme-phoneme model. By following the above procedure, we can extract two sets of English alphabet to romaji edits and their corresponding probabilities. One set is based on direct alignment of Japanese string with English spelling (6) and the other one on the indirect alignment via the English pronunciation as a pivot (7). Although the edits in each set are trained from the same data, the sets are different in the sense that the first set better reflects influence of English spelling on transliteration and the second set better reflects the influence of English pronunciation. Since we would like to obtain a set of edits which reflects the influence both of spelling and pronunciation we combine the two as given in (8). We designate this set as S_{gp} and the corresponding model as GPM.

$$S_g = (\alpha_1, \beta_{11}, P_g(\beta_{11}|\alpha_1)), (\alpha_1, \beta_{12}, P_g(\beta_{12}|\alpha_1)) \ldots (\alpha_n, \beta_{nm}, P_g(\beta_{nm}|\alpha_n)) \tag{6}$$
$$S_p = (\alpha_1, \beta_{11}, P_p(\beta_{11}|\alpha_1)), (\alpha_1, \beta_{12}, P_p(\beta_{12}|\alpha_1)) \ldots (\alpha_p, \beta_{pq}, P_p(\beta_{pq}|\alpha_p)) \tag{7}$$
$$S_{gp} = S_g \cup S_p \qquad \text{s.t. } P_{gp}(\beta_{ij}|\alpha_i) = \gamma P_g(\beta_{ij}|\alpha_i) + \delta P_p(\beta_{ij}|\alpha_i) \quad \text{and} \quad \gamma + \delta = 1 \tag{8}$$

Some statistics of learned mapping sets for $N = 2$ are given in Table 1. We can see that the GM and PM sets differ significantly. Hence, their combination (GPM) results in a more comprehensive set of edits which can handle a wider variety of transliterations.

Table 1. Statistics of learned mapping sets. Statistics given are for romanized Japanese strings

	GM	PM	GPM
Unique English strings	15 120	8 662	18139
Avg. length of English strings (char.)	4.18	4.23	4.34
Max. English string length (char.)	7	11	11
Unique Japanese strings	13 933	7 392	16665
Avg. length of Japanese strings (char.)	4.63	4.66	4.80
Max. Japanese string length (char.)	9	10	10
Japanese strings per English string (avg.)	1.95	4.47	3.35

3.3 Model Implementation

Here, we describe how we calculate back-transliterations for a given input string in romaji. First, the string is segmented as described in [7]. Next, the resulting string is encoded as a Finite State Acceptor (FSA) I in which any path from the start to accept state represents a possible breakup of the string into substring units based on the extracted mapping set S_σ.[8] We add a transition for each substring α_i that appears in the learned mapping set together with special beginning-of-string and end-of-string marks.

Next, we rewrite S_σ as a WFST T with α_i as inputs, β_{ij} as outputs and transition costs as negative logs of probabilities. This WFST T is then inverted and composed with the FSA I to obtain a WFST $I \circ T_I$ which represents all the possible ways to rewrite the input string into English given the mappings learned in the training process.

We also compile the source model into a WFST O so that all word tokens are added as valid outputs. A null input, word delimiter output transition is also added, allowing for multiple words to be output for a single string input. Hence, phrases can also be handled. Note that the valid word tokens need not come from any bilingual/English dictionary but can be any list of words/phrases we would like to make the target of back-transliteration. Probabilities assigned to each token can be either reflecting corpus trained frequencies or uniformly distributed. Thus, the source model can easily be adjusted to the given domain or application.

Finally, WFST $I \circ T_I$ is composed with the WFST O and the resulting WFST $I \circ T_I \circ O$ is searched for k-best transliterations using the k-best path algorithm. A probability $P(E_a|J_a)$ is associated with each path obtained. In cases several paths correspond to the same word (phrase), their probabilities are summed up. Thanks to the cascading WFST composition we are able to search all possible back-transliterations based on the available mappings and select the optimal solution.

[8] Here, $\sigma \in \{g, p, gp\}$.

Table 2. Examples of the NTCIR-2 data

Katakana	Romanized	Segmented	English
ウィンドウサイズ	`uindousaizu`	`uindou#saizu`	*window size*
エキシマレーザ	`ekishimareeza`	`ekishima#reeza`	*excimer laser*
ガスレーザ	`gasureeza`	`gasu#reeza`	*gas laser*
グラビトン	`gurabiton`	`gurabiton`	*graviton*
グリコカリックス	`gurikokarikkusu`	`gurikokarikkusu`	*glycocalyx*

4 Evaluation

We evaluate various aspects of the proposed method on sets of novel katakana strings not in the EDICT dictionary [20]. The first set consists of 150 katakana words extracted from the EDR Japanese corpus [21]. The second test comes from the NTCIR-2 test collection [22]. All 78 out-of-vocabulary katakana words from the topic section (49 short documents) were used. Several examples from this test set are shown in Table 2.

A collection of about 6,000 words in katakana together with the corresponding English translation extracted from the EDICT dictionary was used as training data. This set was expanded, so that for each katakana word containing a long vowel or a geminate consonant, we add one with these removed. The pronunciations for training the PM were obtained from the CMU pronouncing dictionary [23]. When no pronunciations were available the words were excluded from the training. The AT&T FSM library [24] was used for WFST manipulation.

For the EDR test set we used the complete CMU dictionary word set (around 120,000 words) compiled into a language model with word probabilities reflecting the corpus frequencies from the EDR English corpus [21]. For the NTCIR-2 test set we created a language model from about 110,000 words and their frequencies as counted in the English part of the NTCIR-2 collection. The transliterations were considered correct, if they matched the English translation, letter-for-letter, in a non-case-sensitive manner. Table 3 gives results for grapheme-model (GM), phoneme-model (PM), interpolated combination model (COMB) [7] and proposed method of direct combination (GPM).[9] Each model was evaluated with (+SEG) and without segmentation preprocessing module.

We can see that combination generally helps and that in all top-1 and most top-10 cases (regardless of the segmentation), GPM fares better than COMB method. Thus, direct combination seems to be an effective way of combining grapheme- and phoneme-based models. Only for the NTCIR-2 test set, the highest top-1 accuracy is still achieved by GM+SEG model. This can be attributed to a high number of scientific terms whose transliteration better reflects original spelling than pronunciation (e.g. グリコカリックス “`gurikokarikkusu`” *glycocalyx*) that are pushed lower in the combined result sets.

[9] For these experiments the combination weights are arbitrarily set at $\delta = \gamma = 0.5$ for the GPM whereas for the COMB model we use the trained values of δ and γ [7]. GM and PM models were trained with context $N = 2$.

Table 3. Transliteration results for the EDR test set (150 inputs) and for the NTCIR-2 test set (78 inputs)

	EDR test set		NTCIR-2 test set	
	Top-1 (%)	Top-10 (%)	Top-1 (%)	Top-10 (%)
SEG	33.33	34.00	25.64	25.64
GM	48.00	66.00	48.72	62.82
GM+SEG	58.00	72.00	**66.67**	78.21
PM	46.00	61.33	35.90	52.56
PM+SEG	57.33	68.00	57.69	74.36
COMB	49.33	72.00	44.87	67.95
COMB+SEG	59.33	75.33	62.82	**83.33**
GPM	54.00	70.00	48.72	70.51
GPM+SEG	**60.67**	**79.33**	62.82	78.21

4.1 Evaluation on Chinese Test Data

In order to test whether our proposed method is also effective for other languages, we conducted an additional evaluation of back-transliteration on Chinese transliterations of foreign names. The data set used consists of 11,584 transliteration pairs listed by Xinhua News Agency [25]. We use 8,688 pairs to train the system and 2,896 pairs for the evaluation. The training procedure is identical to the one used for Japanese data with the exception of the romanization module which was modified to work with Chinese input and output pinyin transcriptions. We wanted to see whether tone marks and Chinese character (hanzi) segmentation affect the back-transliteration so we compared three different schemes: 1) **Toneless** where pinyin tone marks are removed from the training data, 2) **Tone** where pinyin tone marks are left as-is and **Hanzi** where unit-segmentation is enforced so that all alphabet characters corresponding to one Chinese character are in one unit. For example, for the transliteration 套戦 "`pei4li3`" of *Perry*, we would get the training pairs: (*perry*, `peili`), (*perry*, `pei4li3`) and (*perry*, `pei li`), respectively. Finally, a language model was constructed from all 11,584 words assigned equal weights.

The results of the experiment are given in Table 4. Besides the proposed method, we give the accuracy figures for 3-gram TM, a method proposed by [10] for the same test set (taken from their paper). We can see that for this data set the GM model generally performs better than PM,[10] but that combined models (COMB and GPM) achieve better accuracy than either of the individual models for all three experiment settings. The best performance is achieved by the GPM model trained on pinyin with tone marks, showing that not only is the modeling of pronunciation and spelling simultaneously beneficial but that the direct combination yields better results than interpolation. Finally, we can see that the segmentation negatively affects the top-1 performance in most test

[10] Somewhat worse performance of the PM model can be attributed to a smaller training set due to a large number of pronunciations missing from the CMU dictionary.

Table 4. Transliteration results for the Chinese test set (2,896 inputs)

	Toneless		Tone		Hanzi	
	Top-1 (%)	Top-10 (%)	Top-1 (%)	Top-10 (%)	Top-1 (%)	Top-10 (%)
3-gramTM	N/A	N/A	N/A	N/A	37.90	75.40
GM	68.99	95.99	70.79	94.30	52.14	67.09
GM+SEG	67.68	95.99	70.20	94.30	53.66	70.79
PM	62.81	91.71	65.68	91.69	41.95	62.40
PM+SEG	58.18	91.64	61.11	91.64	40.74	68.78
COMB	69.33	96.82	70.96	96.44	58.49	81.32
COMB+SEG	67.85	96.82	69.99	96.44	57.91	81.32
GPM	69.33	96.75	**72.76**	**96.96**	57.32	80.35
GPM+SEG	68.09	96.75	71.51	**96.96**	58.77	85.08

cases. This can be attributed to the fact that all the inputs in this data set are single words, thus any additional segmentation taxes performance.

4.2 Discussion

For Japanese evaluation sets, we observe similar trends as [7]. The performance of singleton models (GM,PM) can be significantly improved through combination and addition of segmentation module. However, the proposed combination model (GPM) performs better than the interpolated combination (COMB) for most of the test cases/settings. This leads us to believe that a back-transliteration system using direct combination is more robust than systems based on the singleton models or interpolated combination. Although we do not provide evaluation results, we have noticed that increase in the number of tokens in the source model negatively affects both the transliteration speed and accuracy. Thus, it would be beneficial to explore methods for reducing the size of the source model depending on the input and/or domain.

For Chinese evaluation set, we compare our back-transliteration model with 3-gram TM which outperformed other Chinese back-transliteration systems in evaluation given in [10]. 3-gram TM is a character-based model possibly outputting non-valid English strings. Thus, its accuracy could be increased by filtering the outputs against a list of valid tokens. However, such filtering is not an integral part of 3-gram TM. Furthermore, 3-gram TM forces Chinese string segmentation on Chinese character boundaries (akin to the **Hanzi** scheme above) and our experiments show that better results can be achieved by allowing finer segmentation.[11] Although both models consider source and target string context simultaneously, our model considers both the preceding and following context while 3-gram TM model only considers preceding context. Furthermore, our

[11] Admittedly, our model does not incorporate smoothing and is more susceptible to the data sparseness problem that arises when using this alignment scheme. Also [10] use the full set of 34,777 transliteration pairs for training as opposed to 8,688 pair subset we used.

model considers both pronunciation and spelling of the original string but 3-gram TM model only considers the spelling of the original. Given all this, it is not surprising that our model achieves significantly higher accuracy in the evaluation.

5 Conclusion

In this paper we propose a method for improving the back-transliteration accuracy by directly combining grapheme-based and phoneme-based information. Rather than producing back-transliterations based on grapheme and phoneme model independently and then interpolating the results as was previously proposed, we first combine the sets of allowed rewrites (i.e. edits) based on the two models and then calculate the back-transliterations using the combined set. We evaluate the proposed combination method on Japanese transliterations and show that the manner in which grapheme-based and phoneme-based information are combined can significantly affect the system performance.

Furthermore, we show the proposed method can easily be applied to back-transliteration of Chinese and that significant improvements can also be achieved by combining the grapheme and phoneme models.

Acknowledgments. We would like to thank Zhang Min for help with Chinese evaluation data and an anonymous reviewer for valuable comments.

References

1. Knight, K., Graehl, J.: Machine transliteration. Computational Linguistics **24** (1998) 599–612
2. Fujii, A., Ishikawa, T.: Japanese/English cross-language information retrieval: Exploration of query translation and transliteration. Computers and Humanities **35** (2001) 389–420
3. Lin, W.H., Chen, H.H.: Backward machine transliteration by learning phonetic similarity. In: Proc. of the Sixth Conference on Natural Language Learning. (2002) 139–145
4. Stalls, B.G., Knight, K.: Translating names and technical terms in Arabic text. In: Proc. of the COLING/ACL Workshop on Computational Approaches to Semitic Languages. (1998)
5. Jeong, K.S., Myaeng, S.H., Lee, J.S., Choi, K.S.: Automatic identification and back-transliteration of foreign words for information retrieval. Information Processing and Management **35** (1999) 523–540
6. Kang, B.J., Choi, K.S.: Effective foreign word extraction for Korean information retrieval. Information Processing and Management **38** (2002) 91–109
7. Bilac, S., Tanaka, H.: A hybrid back-transliteration system for Japanese. In: Proc. of the 20th International Conference on Computational Linguistics (COLING 2004). (2004) 597–603
8. Kang, B.J., Choi, K.S.: Automatic transliteration and back-transliteration by decision tree learning. In: Proc. of the Second International Conference on Language Resources and Evaluation. (2000)

9. Goto, I., Kato, N., Uratani, N., Ehara, T.: Transliteration considering context information based on the maximum entropy method. In: Proc. of the IXth MT Summit. (2003)
10. Li, H., Zhang, M., Su, J.: A joint source-channel model for machine transliteration. In: Proc. of the 42th Annual Meeting of the Association for Computational Linguistics. (2004) 159–166
11. Brill, E., Kacmarcik, G., Brockett, C.: Automatically harvesting katakana-English term pairs from search engine query logs. In: Proc. of the Sixth Natural Language Processing Pacific Rim Symposium. (2001) 393–399
12. Brill, E., Moore, R.C.: An improved error model for noisy channel spelling correction. In: Proc. of the 38th Annual Meeting of the Association for Computational Linguistics. (2000) 286–293
13. Damerau, F.: A technique for computer detection and correction of spelling errors. Communications of the ACM **7** (1964) 659–664
14. Levenshtein, V.: Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics–Doklady **10** (1966) 707–710
15. Oh, J.H., Choi, K.S.: An English-Korean transliteration model using pronunciation and contextual rules. In: Proc. of the 19th International Conference on Computational Linguistics. (2002) 758–764
16. Eppstein, D.: Finding the k shortest paths. In: Proc. of the 35th Symposium on the Foundations of Computer Science. (1994) 154–165
17. Bilac, S., Tanaka, H.: Improving back-transliteration by combining information sources. In: Proc. of the First International Joint Conference on Natural Language Processing (IJCNLP-04). (2004) 542–547
18. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete via the EM algorithm. Journal of the Royal Statistical Society **39** (1977) 1–38
19. Pereira, F.C.N., Riley, M.: Speech recognition by composition of weighted finite automata. In Roche, E., Shabes, Y., eds.: Finite-State Language Processing. MIT Press (1997) 431–453
20. Breen, J.: EDICT Japanese/English dictionary file (2003) Available: ftp://ftp.cc.monash.edu.au/pub/nihongo.
21. EDR: EDR Electronic Dictionary Technical Guide. Japan Electronic Dictionary Research Institute, Ltd. (1995) (In Japanese).
22. Kando, N., Kuriyama, K., Yoshioka, M.: Overview of Japanese and English Information Retrieval Tasks (JEIR) at the Second NTCIR Wordshop. In: Proc. of NTCIR Workshop 2. (2001)
23. Carnegie Mellon University: The CMU pronouncing dictionary (1998) Available: http://www.speech.cs.cmu.edu/cgi-bin/cmudict.
24. Mohri, M., Pereira, F.C.N., Riley, M.: AT&T FSM library (2003) Available: http://www.research.att.com/ mohri/fsm.
25. Xinhua News Agency: Chinese transliteration of foreign personal names. The Commercial Press (1992)