A hybrid back-transliteration system for Japanese

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

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

Abstract

Transliterating words and names from one language to another is a frequent and highly productive phenomenon. Transliteration is information losing since important distinctions are not preserved in the process. Hence, automatically converting transliterated words back into their original form is a real challenge. In addition, due to its wide applicability in MT and CLIR, it is an interesting problem from a practical point of view. In this paper, we propose a new method, combining the transliterated string segmentation module with phoneme-based and grapheme-based transliteration modules in order to enhance the back–transliterations of Japanese words. Our experiments show significant improvements achieved by the hybrid approach.

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. This process of acquisition and assimilation of a new word into an existing writing system is referred to as transliteration (Knight and Graehl, 1998). For example, the English word *cache* is transliterated in Japanese as キャッシュ “kyasshu”.[1]

Since integration of new words is a very productive process, it often happens that the new pairs are not recorded in machine or human dictionaries. Therefore, it is impossible to rely on the 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 the goal is to retrieve all related documents in two or more languages (Lin and Chen, 2002).

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 preserved when the word is transliterated into the target language. For example, Japanese has only five basic vowels and no /θ/ or /ð/[2] 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”.[3] The system trying to obtain a back-transliteration for バス has therefore three valid choices which cannot be disambiguated in the absence of additional contextual transformation. While the transliteration commonly reflects the pronunciation of the source language, spelling can also affect it. For example the first *e* in *eternal* is transliterated as ”e” instead of ”i” in エターナル “etaanaru”.

Transliterated words are normally written in katakana, one of three Japanese writing systems. While other vocabulary (i.e. animal names or onomatopoeic expressions) can also be written in katakana, the very fact that something is written in katakana is generally a good hint that it might be a transliterated foreign

[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 (Knight and Graehl, 1998), thus closely reflecting English–like pronunciation with long vowels transcribed as ”aa” rather than ”ā”.

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

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

word or a name. Thus, unlike Arabic or Korean, where a big part of the back-transliteration problem is identifying candidate transliterations (Stalls and Knight, 1998; Jeong et al., 1999), in Japanese back-transliteration can be directly applied to any katakana strings absent from the bilingual dictionary.

Furthermore, since Japanese does not employ spaces to delimit words, katakana strings are often not individual English words but whole phrases. For example, リハビリテーションセンター "rihabiriteeshonsentaa" is the transliteration of *rehabilitation center*. Abbreviations are common so the *rehabilitation center* is actually more often transliterated as リハビリセンター "rihabirisentaa".

In this paper we describe a system which first finds the best segmentation of a transliterated string and then obtains back-transliterations using the combined information based on pronunciation and spelling. Our goal is to demonstrate that by recognizing the weaknesses of existing models and carefully combining them to capitalize on their synergies can significantly improve the overall performance even without substantial modifications to the underlying models.

The reminder of this paper is organized as follows: in Section 2 we review previous research. Section 3 describes the proposed segmentation method and Section 4 outlines the transliteration model. Finally, Section 5 gives a short evaluation and a discussion of the results obtained.

2 Previous research

Previous approaches to (back-)transliteration can be roughly divided into two groups: grapheme- and phoneme-based. These approaches are also referred to as direct- and pivot-based methods, respectively.

2.1 Grapheme-based modeling

In this framework, the English string is not converted into a phonemic representation before its alignment with the transliterated string (Kang and Choi, 2000; Goto et al., 2003). Brill et al. (2001) propose a noisy channel model allowing for non-atomics edits (Brill and Moore, 2000). 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 (Damerau, 1964; Levenstein, 1966). This method fails to generate the correct string in cases where English spelling is not reflected in the pronunciation (e.g マイム "maimu" being incorrectly back-transliterated into *maim* instead of *mime*). Furthermore, since the transliterations are compared to dictionary entries this method does not handle phrases directly.

2.2 Phoneme-based modeling

In this approach the pronunciation, rather than the spelling of the original string is considered as a basis for transliteration (Jeong et al., 1999; Oh and Choi, 2002). For Japanese, Knight and Graehl (1998) employ a compositional 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 (Eppstein, 1994). A similar model has been applied for Arabic-English back-transliteration (Stalls and Knight, 1998). However, this model cannot handle cases where the transliteration reflects the original spelling. Furthermore, even though the system handles phrases, commonly it is the case that even though the correct back-transliterations can be obtained for each of the words in the phrase when handled separately, the system does not output the correct answer when handled as a phrase. For example, both テープ "teepu" and サブシステム "sabushisutemu" are correctly transliterated as *tape* and *subsystem*, respectively, but the output for the phrase テープサブシステム "teepusabushisutemu" is incorrect.

3 Input Segmentation

Above we have noted that transliterations are commonly based on English phrases and not only individual words. Therefore, a system able to determine the proper segmentation of the transliterated string W, has a better chance of getting the correct answer. Since Japanese does not employ word delimiters (i.e. white spaces), text segmentation is a big part of any NLP system dealing with Japanese. However, two of the most common morpho-syntactic analysis systems: Juman (Kurohashi et al., 1994) and ChaSen (Matumoto et al., 2002) both employ rule based segmentation which heavily relies on dictionary lookups. Since the back-transliteration is aiming to handle words that are not contained in the dictionary, such systems cannot be readily applied to segmentation of katakana strings. Therefore, statistical segmentation methods are preferable.

Furthermore, the transliterated strings make up a small percentage of Japanese texts, thus making it very hard to obtain a large training set. Thus, the segmentation method must be effective with limited amounts of training data like the algorithms used for word discovery (Brent, 1999; Venkataraman, 2001). Being designed for word acquisition, these methods penalize longer, new words more than shorter ones, hence they often err by over-segmenting the input. However, over-segmentation of the string significantly hurts the transliteration performance since sub-word chunks cannot be transformed to correct English words. In the case of under-segmentation, longer chunks can still be correctly transliterated as phrases (see below). Thus, for transliteration, a system able to segment strings with little training data and without a tendency to over-segment the input string is preferable.

3.1 Segmentation model

Here we outline an implementation of a simple segmentation model which favors longer, new words over shorter ones. Given the string W to be segmented the goal is to insert word delimiters (#) so that the overall score $W = w_1\#w_2\#\ldots\#w_k$ of the string is maximized (Equation 1) if each word w_i score is assigned according to Equations 2 and 3.

$$\hat{W} = \arg\max_{W} \prod_{i=1}^{k} S(w_i) \tag{1}$$

$$S(w_i) = \begin{cases} \frac{C(w_i)}{N+T} & \text{if } C(w_i) > 0 \\ \frac{N}{N+T} S_{\Sigma}(w_i) & \text{otherwise} \end{cases} \tag{2}$$

$$S_{\Sigma}(w_i) = \frac{r(\#)(\prod_{j=1}^{k_i} r(w_i[j]))^p}{1 - r(\#)} \tag{3}$$

$$p = \begin{cases} \bar{L} - k_i & \text{if } k_1 < \bar{L} \\ 1 & \text{otherwise} \end{cases} \tag{4}$$

In these equation C is the occurrence count, N is the count of word types and T is the total token count. k_i is the length of word w_i, $w_i[j]$ is the jth letter, r is the relative frequency and $\bar{L}$ is the average word length. As can be seen, penalty p is introduced for discouraging segmentations introducing many short, novel words (Equation 4). The best segmentation is calculated using Dynamic Programming. Once the best segmentation is determined, the counts are updated as necessary. The penalty for shorter novel words is what distinguishes this model from the unigram model of Venkataraman (2001).[4]

4 Transliteration models

After the input is segmented we can proceed to transliterate it back into original language. As pointed out above, both the pronunciation and spelling of the original influence the transliteration. This being so, we combine them to achieve higher accuracy.

Given some Japanese word in romaji (J_a),[5] 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. We train the channel model as described below, and then reverse it to handle the romaji input.

4.1 Grapheme-based model (GM)

In this model the English word is directly rewritten as a Japanese romaji string with the probability $P_g(J_a|E_a)$. Here, we follow (Brill et al., 2001) to arbitrarily break up the E_a string into n parts and output each part independently. Thus, the resulting probability of outputting J_a can be rewritten as in the equation (5).

$$P_g(J_a|E_a) \cong \prod_{i=1}^{n} P_g(J_{a_i}|E_{a_i}) \tag{5}$$

We implement $P_g(J_a|E_a)$ as a weighted Finite State Transducer (WFST) with E_{a_i} as inputs, J_{a_i} as outputs (Pereira and Riley, 1997; Knight and Graehl, 1998) and transition costs as negative logs of probabilities. This WFST is then reversed and composed with the source model WFST.[6] When the source model $P(E_a)$ is compiled into a WFST, a null input, word delimiter output transition is added, allowing for multiple words to be output for a single string input. Hence phrases can also be handled.

[4] Venkataraman evaluates unigram, bigram and trigram models, often observing the best performance with the unigram model.

[5] As stated above, transliterated words are normally written in katakana, potentially inducing an another stage in the model: rewriting romaji characters into katakana $P(J_k|J_a)$. However, katakana characters generally have a unique alphabetic equivalent, thus reducing this distribution to 1. We implement the katakana to romaji conversion as a preprocessing module.

[6] We use the AT&T FSM library (`http://www.research.att.com/~mohri/fsm/`) for WFST composition.

The WFST resulting from the composition of the $P_g(J_a|E_a)$ and $P(E_a)$ WFST composition is searched for k-best transliterations using the k-best path algorithm. A probability $P_g(E_a|J_a)$ is associated with each path obtained.

4.2 Phoneme-based model (PM)

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 the English phonemes into romaji with some probability $P(J_a|E_p)$. Consequently, $P_p(J_a|E_a)$ can be rewritten as equation (6). Rather than manipulating these two distributions separately, we compute their composition to obtain a unique probability distribution $P_p(J_{a_i}|E_{a_i})$.

$$P_p(J_a|E_a) \cong \prod_{i=1}^{n} P(J_{a_i}|E_{p_i}) \times \prod_{i=1}^{n} P(E_{p_i}|E_{a_i}) \quad (6)$$

Consequently 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]

4.3 Combining the models

After obtaining the back–transliterations $E_{a_{phon}}$ and $E_{a_{graph}}$ with the respective probabilities of $P_p(E_a|J_a)$ and $P_g(E_a|J_a)$, we can assign the final score of a transliteration $S_c(E_a|J_a)$ as in equation (7) where γ and δ are set to maximize the accuracy on the training set.[8] Transliteration with the highest score is selected as the best.

$$\begin{aligned} S_c(E_a|J_a) &= \gamma P_p(E_a|J_a) + \delta P_g(E_a|J_a) \\ &\quad \text{s.t.} \quad \gamma + \delta = 1 \end{aligned} \quad (7)$$

4.4 Training the models

For the GM, we follow (Brill et al., 2001) closely to extract the character-string mappings. Romaji and English alphabet are first aligned using the non-weighted Levensthein distance. Then, letter-edits are expanded to include up to N edits to the right and to the left. For example, for the pair (`roo`,*row*) we get: $r \rightarrow r \quad o \rightarrow o \quad o \rightarrow w$. For $N = 1$, edits $ro \rightarrow ro$, $roo \rightarrow row$, $oo \rightarrow ow$ are also added to the set. We collect a complete set of edits $\alpha_g \rightarrow \beta_g$ in the training set and assign the probability to each according to equation (8). Throughout, we distinguish edits that appear at the beginning or the end of the word or neither.

$$P(\alpha \rightarrow \beta) = \frac{count(\alpha \rightarrow \beta)}{count(\alpha)} \quad (8)$$

Given the collection of edits $\alpha_g \rightarrow \beta_g$ for each input word J_a we can generate a WFST which contains all possible ways to rewrite the input string.

For the PM, we follow (Knight and Graehl, 1998) to obtain the optimal romaji to English phoneme alignment. After the EM algorithm 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 $\alpha_{e_p} \rightarrow \beta_{j_a}$. This process is repeated to obtain the set of all possible rewrites of English alphabet into phonemes $\alpha_{e_a} \rightarrow \beta_{e_p}$.

Each input α_{e_a} with all its mappings β_{e_p} is converted into a WFST and composed with a WFST encoding the complete set of mappings $\alpha_{e_p} \rightarrow \beta_{j_a}$ to obtain the set of all possible rewrites of English alphabet strings α_p into romaji strings β_p based on the PM.

5 Evaluation

The proposed system is intended for producing transliterations of katakana strings not found in the system dictionary. Therefore, we evaluate various aspects of the proposed method on sets of novel katakana strings not in the EDICT dictionary.[9] The first set consists of 150 katakana words extracted from the EDR Japanese corpus (EDR, 1995). To examine possible uses of transliteration in CLIR, we conducted a second test using the NTCIR-2 test collection (Kando et al., 2001). All 78 out-of-vocabulary katakana words from the topic section (i.e. queries) were used. Note that topics section consists of 49 short documents, thus showing that NTCIR-2 collection of scientific texts has a large number of (novel) katakana words.

First, we evaluate the segmentation module of the system. We use the two sets and compare our system (SEG) with the unigram segmentation model of Venkataraman (2001) (UNI), and the segmentation obtained by ChaSen (CHA).[10]

[7] However, the pronunciation dictionary is still necessary for the training.

[8] Parameters are trained using Golden Section Search (Press et al., 1992).

[9] `ftp://ftp.cc.monash.edu.au/pub/nihongo/`

[10] Only given for NTCIR-2 data since EDR katakana words with frequency over 3 were used for evaluation.

	Set	Recall	Precision	F
UNI	EDR	78.64	65.53	71.49
SEG	EDR	**95.45**	**95.00**	**95.23**
CHA	NTCIR-2	65.04	66.67	65.84
UNI	NTCIR-2	82.11	70.14	75.65
SEG	NTCIR-2	**88.62**	**86.51**	**87.56**

Table 1: Results of the segmentation

	Top-1 (%)	Top-10 (%)
PM0	20 (13.33)	36 (24.00)
SEG	50 (33.33)	51 (34.00)
GM	72 (48.00)	99 (66.00)
GM + SEG	87 (58.00)	108 (72.00)
PM	69 (46.00)	92 (61.33)
PM + SEG	86 (57.33)	102 (68.00)
COMB	74 (49.33)	108 (72.00)
COMB + SEG	89 (**59.33**)	113 (**75.33**)

Table 2: Transliteration results for the EDR test set (150 inputs)

For this evaluation we trained the two statistical models on a complete set of about 13,000 all-katakana strings from the EDICT dictionary. Each string was considered as one word and the unigram and phoneme models were updated after each repetition. The segmentation standard was segmented by hand.

The results are shown in Table 1. In this table, recall is calculated as $\frac{c}{N}$, precision as $\frac{c}{n}$ and F-measure as $\frac{2\times precision\times recall}{precision+recall}$. Here, N is the correct number of words (assigned in hand-segmentation), e is the number of words incorrectly identified, c is the number of words correctly identified and $n = c + e$ is the total number of words identified automatically. We can see that ChaSen segmentation achieves the lowest score of the three methods because it deems most strings as unknown words and leaves them unsegmented. UNI method performs better, but it assigns a high number of incorrect word boundaries. The SEG method is highly effective in encouraging selection of longer novel words by penalizing shorter words. This reduces the number of segmentation errors and significantly increases all three measures.

Next we evaluate the transliteration module of the system. We extracted a collection of about 6000 words in katakana together with the corresponding English translation from the EDICT dictionary. 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.[11] When no pronunciations were available the words were excluded from the training. The parameters were tuned on a different 700+ word subset of the EDICT dictionary.

	Top-1 (%)	Top-10 (%)
SEG	20 (25.64)	20 (25.64)
GM	38 (48.72)	49 (62.82)
GM + CHA	42 (53.85)	54 (69.23)
GM + UNI	42 (53.85)	52 (66.67)
GM + SEG	52 (**66.67**)	61 (78.21)
PM	28 (35.90)	41 (52.56)
PM + CHA	38 (48.72)	49 (62.82)
PM + UNI	39 (50.00)	55 (70.51)
PM + SEG	45 (57.69)	58 (74.36)
COMB	35 (44.87)	53 (67.95)
COMB + CHA	41 (52.56)	56 (71.79)
COMB + UNI	42 (53.85)	60 (76.92)
COMB + SEG	49 (62.82)	65 (**83.33**)

Table 3: Transliteration results for the NTCIR-2 test set (78 inputs)

In the first experiment 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 (EDR, 1995) and look for transliterations of 150 words in the EDR test set. The transliterations were considered correct, if they matched the English translation, letter-for-letter, in a non-case-sensitive manner.

Table 2 gives the transliteration results respectively for the Phoneme Model without context (PM0), the proposed segmentation with dictionary lookup (SEG), the Grapheme Model (GM), the Phoneme Model (PM) and the combined model (COMB) and the combinations of the latter three with the SEG model. When the SEG model is used, the input string is first segmented and then a dictionary lookup is performed on each segment. When this model is combined with other models, the translitera-

The ChaSen segmentation depends on the sentence context, hence it might be different for different appearances of the string.

[11] http://www.speech.cs.cmu.edu/cgi-bin/cmudict

tion is produced for each segment regardless of whether dictionary lookup was successful or not. This is necessary because dictionary information often does not correspond to the desired transliteration (e.g. マルチ "maruchi" is translated only as *multimedia* in EDICT, although it would often be better translated as *multi*). The PM0 was trained only on the directly aligning edits (N=0), and the PM and GM models used a context of two units to the left and to the right (N=2).

We can see that segmenting the string improves the performance of any model it is combined with, and that the GM and PM models achieve similar results when equivalent context is used. However, the set of correctly handled entries is different for each one of them, hence their combination increases coverage. Overall, segmentation with the combined grapheme and phoneme models results in the best transliteration accuracy. Among others, this model successfully handles phrases with abbreviations (e.g. レハビリセンター "rehabirisentaa" is correctly transliterated into *rehabilitation center*) provided the abbreviated constituent is recorded in the dictionary.

In the second experiment we created a dictionary model from about 110,000 words and their frequencies as counted in the English part of the NTCIR-2 collection and produced transliterations of the 78 novel katakana words. We were also interested in evaluating the influence of segmentation accuracy on the transliteration so we give results for two additional segmentation methods: segmentation by ChaSen (CHA) and Venkataraman's unigram model (UNI). The results are given in Table 3. We can see that any segmentation method improves the overall performance, but the largest increase in accuracy is achieved by using the proposed segmentation. Note that the $GM + SEG$ model has best top-1 accuracy for this data set, but the best top-10 accuracy is still achieved by $COMB + SEG$ model. This is due 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 result set when the transliteration scores are interpolated. Still, the need to consider both spelling and pronunciation simultaneously is reinforced, since which of the two transliteration models (i.e. PM or GM) performs better depends on the input string.

5.1 Discussion

Brill et al. (2001) provide no direct evaluation of their transliteration system. Instead, they evaluate the ability of their system to extract English-katakana pairs from non-aligned web query logs. Furthermore, no mention is made of handling English phrases.

On the other hand, Knight and Graehl (1998) give only the accuracy for transliteration of personal names (64% correct, 12% phonetically equivalent), but not for general out-of-vocabulary terms. Their system does handle English phrases, but uses no context information. Also, it uses pronunciation only of the most frequent words.

Furthermore, there is no common test set for evaluation of back-transliteration. All this makes comparison with our system difficult.

Nonetheless, our GM corresponds to Brill's system, augmented with phrase handling ability. On the other hand, the PM0 is similar to Knight's system, except that we do not model romaji to katakana ambiguities. Also, we combine mappings from English spelling to pronunciation and pronunciation to romaji into a single WFST rather than handling them separately in the transliteration process.[12] The results obtained show the advantages of the proposed approach and there is no reason to believe that improvements in any of the individual modules would not be reflected in the overall accuracy of the combined system.

Based on our observations of the weaknesses of previous systems we were able to make small improvements to each one of the modules. The resulting system, combining all these modules achieved significant overall improvement. This shows that, rather than trying to develop a highly complex model able to handle all aspects of the problem at hand, sometimes it is effective enough to combine the strengths of smaller simpler models in order to improve the system's overall performance.

In the future, we would like to explore in more depth the effect of the improved transliteration on CLIR systems. We would also like to study the benefits of using context available in CLIR queries on the transliteration accuracy.

[12] It is possible that in the PM0 case these changes have reduced the system accuracy. Indeed Knight and Graehl (1998) mention slightly better performance when mapping English spelling to pronunciation is not modeled probabilistically.

6 Conclusion

Back transliteration is the process of converting transliterated words back into their original form. Previous models used either only phoneme- or grapheme-based information contained in the transliteration. Furthermore, no segmentation of input strings is performed. In this paper we show that the performance of back-transliteration can be significantly improved by augmentations of the phoneme-based, grapheme-based transliteration and the segmentation modules and their combination into a hybrid system. The system evaluation on two sets of transliterated strings, not contained in the system dictionary, shows significant increase in accuracy over singleton models.

Acknowledgments

We would like to thank Timothy Baldwin, Taiichi Hashimoto, Takenobu Tokunaga, Michael Zock and two anonymous reviewers on valuable comments and help in writing this paper.

References

M. R. Brent. 1999. An efficient, probabilistically sound algorithm for segmentation and word discovery. *Machine Learning*, 34:71–105.

E. Brill and R. C. Moore. 2000. An improved error model for noisy channel spelling correction. In *Proc. of ACL2000*, pages 286–293, Tokyo, Japan.

E. Brill, G. Kacmarcik, and C. Brockett. 2001. Automatically harvesting katakana-English term pairs from search engine query logs. In *Proc. of NLPRS01*, pages 393–399, Tokyo, Japan.

F. Damerau. 1964. A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7:659–664.

EDR. 1995. *EDR Electronic Dictionary Technical Guide*. Japan Electronic Dictionary Research Institute, Ltd. (In Japanese).

D. Eppstein. 1994. Finding the k shortest paths. In *Proc. of the 35th Symposium on the Foundations of Computer Science*, pages 154–165.

I. Goto, N. Kato, N. Uratani, and T. Ehara. 2003. Transliteration considering context information based on the maximum entropy method. In *Proc. of IXth MT Summit*.

K. S. Jeong, S. H. Myaeng, J. S. Lee, and K. S. Choi. 1999. Automatic identification and back-transliteration of foreign words for information retrieval. *Information Processing and Management*, 35:523–540.

Noriko Kando, Kazuko Kuriyama, and Masaharu Yoshioka. 2001. Overview of Japanese and English Information Retrieval Tasks (JEIR) at the Second NTCIR Wordshop. In *Proc. of NTCIR Workshop 2*.

B. J. Kang and K. S. Choi. 2000. Automatic transliteration and back-transliteration by decision tree learning. In *Proc. of LREC2000*.

K. Knight and J. Graehl. 1998. Machine transliteration. *Computational Linguistics*, 24:599–612.

S. Kurohashi, T. Nakamura, Y. Matsumoto, and M. Nagao. 1994. Improvements of Japanese morphological analyzer JUMAN. In *SNLR*, pages 22–28.

V. Levensthein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics–Doklady*, 10:707–710.

W. H. Lin and H. H. Chen. 2002. Backward machine transliteration by learning phonetic similarity. In *Proc. of CoNLL2002*, pages 139–145.

Y. Matumoto, A. Kitauchi, T. Yamashita, Y. Hirano, H. Matsuda, K. Takaoka, and M. Asahara. 2002. Morphological analysis system ChaSen version 2.2.9 manual.

J. H. Oh and K. S. Choi. 2002. An English-Korean transliteration model using pronunciation and contextual rules. In *Proc. of COLING2002*, pages 393–399.

F. C. N. Pereira and M. Riley. 1997. Speech recognition by composition of weighted finite automata. In E. Roche and Y. Shabes, editors, *Finite-State Language Processing*, pages 431–453. MIT Press.

W. H. Press, B. P. Flannery, A. Teukolsky, and T. Vetterling. 1992. *Numeric Recipies in C*. Cambridge University Press, 2nd edition.

B. G. Stalls and K. Knight. 1998. Translating names and technical terms in Arabic text. In *Proc. of the COLING/ACL Workshop on Computational Approaches to Semitic Languages*.

A. Venkataraman. 2001. A statistical model for word discovery in transcribed speech. *Computational Linguistics*, 27:352–372.