Back Transliteration from Japanese to English Using Target English Context

Isao Goto[†], Naoto Kato[††], Terumasa Ehara[†††], and Hideki Tanaka[†]

[†]NHK Science and Technical Research Laboratories
1-11-10 Kinuta, Setagaya, Tokyo, 157-8510, Japan
goto.i-es@nhk.or.jp
tanaka.h-ja@nhk.or.jp

[††]ATR Spoken Language Translation Research Laboratories
2-2-2 Hikaridai, Keihanna Science City, Kyoto, 619-0288, Japan
naoto.kato@atr.jp

[†††]Tokyo University of Science, Suwa
5000-1, Toyohira, Chino, Nagano, 391-0292, Japan
eharate@rs.suwa.tus.ac.jp

Abstract

This paper proposes a method of automatic back transliteration of proper nouns, in which a Japanese transliterated-word is restored to the original English word. The English words are created from a sequence of letters; thus our method can create new English words that are not registered in dictionaries or English word lists. When a katakana character is converted into English letters, there are various candidates of alphabetic characters. To ensure adequate conversion, the proposed method uses a target English context to calculate the probability of an English character or string corresponding to a Japanese katakana character or string. We confirmed the effectiveness of using the target English context by an experiment of personal-name back transliteration.

1 Introduction

In transliteration, a word in one language is converted into a character string of another language expressing how it is pronounced. In the case of transliteration into Japanese, special characters called katakana are used to show how a word is pronounced. For example, a personal name and its transliterated word are shown below.

[Transliteration]
Cunningham ⇨ カニンガム
(*ka ni n ga mu*)

Here, the italic alphabets are romanized Japanese katakana characters.

New transliterated words such as personal names or technical terms in katakana are not always listed in dictionaries. It would be useful for cross-language information retrieval if these words could be automatically restored to the original English words.

Back transliteration is the process of restoring transliterated words to the original English words. Here is a problem of back transliteration.

[Back transliteration]
? ⇦ クラッチフィールド
(English word) (*ku ra cchi fi – ru do*)

There are many ambiguities to restoring a transliterated katakana word to its original English word. For example, should "*a*" in "*ku ra cchi fi – ru do*" be converted into the English letter of "a" or "u" or some other letter or string? Trying to resolve the ambiguity is a difficult problem, which means that back transliteration to the correct English word is also difficult.

Using the pronunciation of a dictionary or limiting output English words to a particular English word list prepared in advance can simplify the problem of back transliteration. However, these methods cannot produce a new English word that is not registered in a dictionary or an English word list. Transliterated words are mainly proper nouns and technical terms, and such words are often not registered. Thus, a back transliteration framework for creating new words would be very useful.

A number of back transliteration methods for selecting English words from an English pronunciation dictionary have been proposed. They include Japanese-to-English (Knight and Graehl, 1998)[1], Arabic-to-English (Stalls and Knight,

[1] Their English letter-to-sound WFST does not convert English words that are not registered in a pronunciation dictionary.

1998), and Korean-to-English (Lin and Chen, 2002).

There are also methods that select English words from an English word list, e.g., Japanese-to-English (Fujii and Ishikawa, 2001) and Chinese-to-English (Chen et al., 1998).

Moreover, there are back transliteration methods capable of generating new words, there are some methods for back transliteration from Korean to English (Jeong et al., 1999; Kang and Choi, 2000).

These previous works did not take the target English context into account for calculating the plausibility of matching target characters with the source characters.

This paper presents a method of taking the target English context into account to generate an English word from a Japanese katakana word. Our character-based method can produce new English words that are not listed in the learning corpus.

This paper is organized as follows. Section 2 describes our method. Section 3 describes the experimental set-up and results. Section 4 discusses the performance of our method based on the experimental results. Section 5 concludes our research.

2 Proposed Method

2.1 Advantage of using English context

First we explain the difficulty of back transliteration without a pronunciation dictionary. Next, we clarify the reason for the difficulty. Finally, we clarify the effect using English context in back transliteration.

In back transliteration, an English letter or string is chosen to correspond to a katakana character or string. However, this decision is difficult. For example, there are cases that an English letter "u" corresponds to "*a*" of katakana, and there are cases that the same English letter "u" does not correspond to the same "*a*" of katakana. "u" in Cunningham corresponds to "*a*" in katakana and "u" in Bush does not correspond to "*a*" in katakana. It is difficult to resolve this ambiguity without the pronunciation registered in a dictionary.

The difference in correspondence mainly comes from the difference of the letters around the English letter "u." The correspondence of an English letter or string to a katakana character or string varies depending on the surrounding characters, i.e., on its English context.

Thus, our back transliteration method uses the target English context to calculate the probability of English letters corresponding to a katakana character or string.

2.2 Notation and conversion-candidate lattice

We formulate the word conversion process as a unit conversion process for treating new words. Here, the unit is one or more characters that form a part of characters of the word.

A katakana word, $\boldsymbol{K}$, is expressed by equation 2.1 with "^" and "$" added to its start and end, respectively.

$$\boldsymbol{K} = k_0^{m+1} = k_0 k_1 ... k_{m+1} \quad (2.1)$$

$$k_0 = \text{^}, \quad k_{m+1} = \$ \quad (2.2)$$

where k_j is the *j*-th character in the katakana word, and *m* is the number of characters except for "^" and "$" and k_0^{m+1} is a character string from k_0 to k_{m+1}.

We use katakana units constructed of one or more katakana characters. We denote a katakana unit as ***ku***. For any ***ku***, many English units, ***eu***, could be corresponded as conversion-candidates. The ***ku***'s and ***eu***'s are generated using a learning corpus in which bilingual words are separated into units and every ***ku*** unit is related an ***eu*** unit.

$\mathcal{L}\{\boldsymbol{E}\}$ denotes the lattice of all ***eu***'s corresponding to ***ku***'s covering a Japanese word. Every ***eu*** is a node of the lattice and each node is connected with next nodes. $\mathcal{L}\{\boldsymbol{E}\}$ has a lattice structure starting from "^" and ending at "$." Figure 1 shows an example of $\mathcal{L}\{\boldsymbol{E}\}$ corresponding to a katakana word "キルシュシュタイン(*ki ru shu shu ta i n*)." In the figure, each circle represents one ***eu***.

A character string linking individual character units in the paths $p_d \in (p_1, p_2, .., p_q)$ between "^" and "$" in $\mathcal{L}\{\boldsymbol{E}\}$ becomes a conversion candidate, where *q* is the number of paths between "^" and "$" in $\mathcal{L}\{\boldsymbol{E}\}$.

We get English word candidates by joining ***eu***'s from "^" to "$" in $\mathcal{L}\{\boldsymbol{E}\}$. We select a certain path, p_d, in $\mathcal{L}\{\boldsymbol{E}\}$. The number of character units

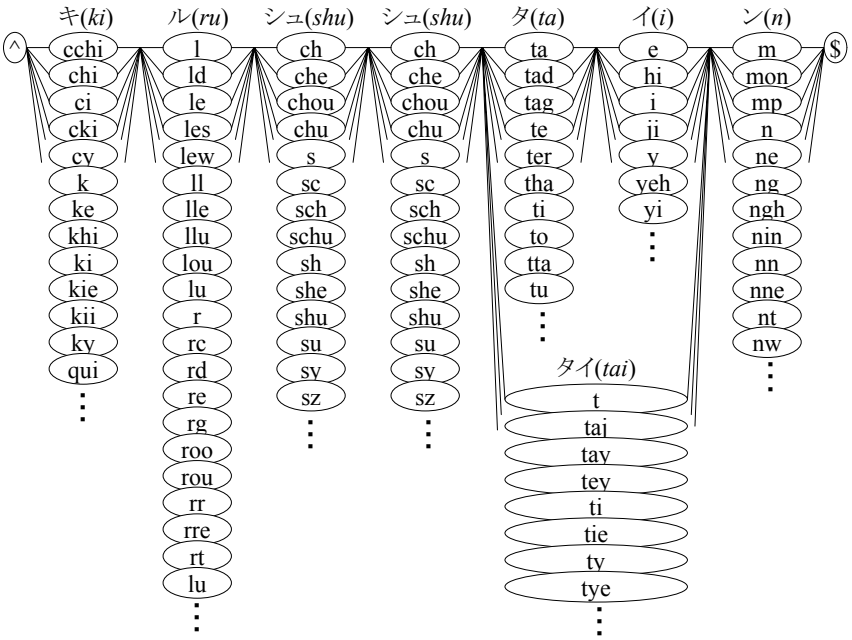

Figure 1: Example of lattice $\mathcal{L}\{\boldsymbol{E}\}$ of conversion candidates units.

except for "^" and "$" in p_d is expressed as $n(p_d)$. The character units in p_d are numbered from start to end.

The English word, $\boldsymbol{E}$, resulting from the conversion of a katakana word, $\boldsymbol{K}$, for p_d is expressed as follows:

$$\begin{aligned}\boldsymbol{K} &= k_0^{m+1} = k_0 k_1 .. k_{m+1} \\ &= \boldsymbol{ku}_0^{n(p_d)+1} = \boldsymbol{ku}_0 \boldsymbol{ku}_1 .. \boldsymbol{ku}_{n(p_d)+1}, \end{aligned} \quad (2.3)$$

$$\begin{aligned}\boldsymbol{E} &= e_0^{l(p_d)+1} = e_0 e_1 .. e_{l(p_d)+1} \\ &= \boldsymbol{eu}_0^{n(p_d)+1} = \boldsymbol{eu}_0 \boldsymbol{eu}_1 .. \boldsymbol{eu}_{n(p_d)+1}, \end{aligned} \quad (2.4)$$

$$k_0 = e_0 = \boldsymbol{ku}_0 = \boldsymbol{eu}_0 = \text{^},$$

$$k_{m+1} = e_{l(p_d)+1} = \boldsymbol{ku}_{n(p_d)+1} = \boldsymbol{eu}_{n(p_d)+1} = \$, \quad (2.5)$$

where e_j is the j-th character in the English word. $l(p_d)$ is the number of characters except for "^" and "$" in the English word. $\boldsymbol{eu}_0^{n(p_d)+1}$ for each p_d in $\mathcal{L}\{\boldsymbol{E}\}$ in equation 2.4 becomes the candidate English word. $\boldsymbol{ku}_0^{n(p_d)+1}$ in equation 2.3 shows the sequence of katakana units.

2.3 Probability models using target English context

To determine the corresponding English word for a katakana word, the following equation 2.6 must be calculated:

$$\hat{\boldsymbol{E}} = \arg\max_{\boldsymbol{E}} P(\boldsymbol{E} \mid \boldsymbol{K}). \quad (2.6)$$

Here, $\hat{\boldsymbol{E}}$ represents an output result.

To use the English context for calculating the matching of an English unit with a katakana unit, the above equation is transformed into Equation 2.7 by using Bayes' theorem.

$$\hat{\boldsymbol{E}} = \arg\max_{\boldsymbol{E}} P(\boldsymbol{E}) P(\boldsymbol{K} \mid \boldsymbol{E}) \quad (2.7)$$

Equation 2.7 contains a translation model in which an English word is a condition and katakana is a result.

The word in the translation model $P(\boldsymbol{K} \mid \boldsymbol{E})$ in Equation 2.7 is broken down into character units by using equations 2.3 and 2.4.

$$\begin{aligned}\hat{\boldsymbol{E}} &= \arg\max_{\boldsymbol{E}} P(\boldsymbol{E}) \\ &\quad \times \sum_{\boldsymbol{eu}_0^{n(p_d)+1}} \sum_{\boldsymbol{ku}_0^{n(p_d)+1}} P(\boldsymbol{K}, \boldsymbol{ku}_0^{n(p_d)+1}, \boldsymbol{eu}_0^{n(p_d)+1} \mid \boldsymbol{E}) \\ &= \arg\max_{\boldsymbol{E}} P(\boldsymbol{E}) \\ &\quad \times \sum_{\boldsymbol{eu}_0^{n(p_d)+1}} \sum_{\boldsymbol{ku}_0^{n(p_d)+1}} \Big\{ P(\boldsymbol{K} \mid \boldsymbol{ku}_0^{n(p_d)+1}, \boldsymbol{eu}_0^{n(p_d)+1}, \boldsymbol{E}) \\ &\qquad \times P(\boldsymbol{ku}_0^{n(p_d)+1} \mid \boldsymbol{eu}_0^{n(p_d)+1}, \boldsymbol{E}) P(\boldsymbol{eu}_0^{n(p_d)+1} \mid \boldsymbol{E}) \Big\} \end{aligned} \quad (2.8)$$

$\boldsymbol{eu}_0^{n(p_d)+1}$ includes information of $\boldsymbol{E}$. $\boldsymbol{K}$ is only affected by $\boldsymbol{ku}_0^{n(p_d)+1}$. Thus equation 2.8 can be rewritten as follows:

$$\hat{\boldsymbol{E}} = \arg\max_{\boldsymbol{E}} P(\boldsymbol{E}) \times \sum_{\boldsymbol{eu}_0^{n(p_d)+1}} \sum_{\boldsymbol{ku}_0^{n(p_d)+1}} \left\{ P(\boldsymbol{K} \mid \boldsymbol{ku}_0^{n(p_d)+1}) \times P(\boldsymbol{ku}_0^{n(p_d)+1} \mid \boldsymbol{eu}_0^{n(p_d)+1}) P(\boldsymbol{eu}_0^{n(p_d)+1} \mid \boldsymbol{E}) \right\}. \quad (2.9)$$

$P(\boldsymbol{K} \mid \boldsymbol{ku}_0^{n(p_d)+1})$ is 1 when the string of $\boldsymbol{K}$ and $\boldsymbol{ku}_0^{n(p_d)+1}$ is the same, and the strings of the $\boldsymbol{ku}_0^{n(p_d)+1}$ of all paths in the lattice and the string of the $\boldsymbol{K}$ is the same. Thus, $P(\boldsymbol{K} \mid \boldsymbol{ku}_0^{n(p_d)+1})$ is always 1.

We approximate the sum of paths by selecting the maximum path.

$$\hat{\boldsymbol{E}} \approx \arg\max_{\boldsymbol{E}} P(\boldsymbol{E}) P(\boldsymbol{ku}_0^{n(p_d)+1} \mid \boldsymbol{eu}_0^{n(p_d)+1}) \times P(\boldsymbol{eu}_0^{n(p_d)+1} \mid \boldsymbol{E}) \quad (2.10)$$

We show an instance of each probability model with a concrete value as follows:

$P(\boldsymbol{E})$
$P(\text{^Crutchfield\$})$,

$P(\boldsymbol{K} \mid \boldsymbol{ku}_0^{n(p_d)+1})$
$P(\text{^クラッチフィールド\$} \mid \text{^/ク/ラ/ッチ/フィー/ル/ド/\$})$,
(*ku ra cchi fi – ru do*) (*ku / ra / cchi / fi – / ru / do*)

$P(\boldsymbol{ku}_0^{n(p_d)+1} \mid \boldsymbol{eu}_0^{n(p_d)+1})$
$P(\text{^/ク/ラ/ッチ/フィー/ル/ド/\$} \mid \text{^/C/ru/tch/fie/l/d/\$})$,
(*ku / ra / cchi / fi – / ru / do*)

$P(\boldsymbol{eu}_0^{n(p_d)+1} \mid \boldsymbol{E})$
$P(\text{^/C/ru/tch/fie/ld/\$} \mid \text{^Crutchfield\$})$.

We broke down the language model $P(\boldsymbol{E})$ in equation 2.10 into letters.

$$P(\boldsymbol{E}) \approx \prod_{j=1}^{l(p_d)+1} P(e_j \mid e_{j-a}^{j-1}) \quad (2.11)$$

Here, a is a constant. Equation 2.11 is an (a+1)-gram model of English letters.

Next, we approximate the translation model $P(\boldsymbol{ku}_0^{n(p_d)+1} \mid \boldsymbol{eu}_0^{n(p_d)+1})$ and the chunking model $P(\boldsymbol{eu}_0^{n(p_d)+1} \mid \boldsymbol{E})$. For this, we use our previously proposed approximation technique (Goto et al., 2003). The outline of the technique is shown as follows.

$P(\boldsymbol{ku}_0^{n(p_d)+1} \mid \boldsymbol{eu}_0^{n(p_d)+1})$ is approximated by reducing the condition.

$$P(\boldsymbol{ku}_0^{n(p_d)+1} \mid \boldsymbol{eu}_0^{n(p_d)+1}) = \prod_{i=1}^{n(p_d)+1} P(\boldsymbol{ku}_i \mid \boldsymbol{ku}_0^{i-1}, \boldsymbol{eu}_0^{n(p_d)+1}) \approx \prod_{i=1}^{n(p_d)+1} P(\boldsymbol{ku}_i \mid e_{start(i)-b}^{start(i)-1}, \boldsymbol{eu}_i, e_{end(i)+1}) \quad (2.12)$$

where $start(i)$ is the first position of the i-th character unit $\boldsymbol{eu}_i$, while $end(i)$ is the last position of the i-th character unit $\boldsymbol{eu}_i$; and b is a constant. Equation 2.12 takes English context $e_{start(i)-b}^{start(i)-1}$ and $e_{end(i)+1}$ into account.

Next, the chunking model $P(\boldsymbol{eu}_0^{n(p_d)+1} \mid \boldsymbol{E})$ is transformed. All chunking patterns of $\boldsymbol{E} = e_0^{l(p_d)+1}$ into $\boldsymbol{eu}_0^{n(p_d)+1}$ are denoted by each $l(p_d)$+1 point between $l(p_d)$+2 characters that serve or do not serve as delimiters. $\boldsymbol{eu}_0$ and $\boldsymbol{eu}_{n(p_d)+1}$ are determined in advance. $l(p_d)$-1 points remain ambiguous. We represent the value that is delimiter or is non-delimiter between e_j and e_{j+1} by z_j. We call the z_j delimiter distinction.

$$z_j = \begin{cases} \text{delimiter} \\ \text{non-delimiter} \end{cases} \quad (2.13)$$

Here, we show an example of English units using z_j.

	$(e_1\ e_2\ e_3\ e_4\ e_5\ e_6\ e_7\ e_8\ e_9\ e_{10}\ e_{11})$
English:	C/ r u/t c h/f i e/l/d
Values of z_j:	1 0 1 0 0 1 0 0 1 1
	$(z_1\ z_2\ z_3\ z_4\ z_5\ z_6\ z_7\ z_8\ z_9\ z_{10})$

In this example, a delimiter of z_j is represented by 1 and a non-delimiter is represented by 0.

The chunking model is transformed into a processing per character by using z_j. And we reduce the condition.

$$P(\boldsymbol{eu}_0^{n(p_d)+1} \mid \boldsymbol{E}) = P(z_0^{l(p_d)-1} \mid e_0^{l(p_d)+1}) = \prod_{j=1}^{l(p_d)-1} P(z_j \mid z_0^{j-1}, e_0^{l(p_d)+1}) \approx \prod_{j=1}^{l(p_d)-1} P(z_j \mid z_{j-c-1}^{j-1}, e_{j-c}^{j+1}) \quad (2.14)$$

The conditional information of the English e_{j-c}^{j+1} is as many as c characters and 1 character before and after z_j, respectively. The conditional information of z_{j-c-1}^{j-1} is as many as c+1 delimiter distinctions before z_j.

By using equation 2.11, 2.12, and 2.14, equation 2.10 becomes as follows:

$$\hat{\boldsymbol{E}} \approx \arg\max_{\boldsymbol{E}} \prod_{j=1}^{l(p_d)+1} P(e_j \mid e_{j-a}^{j-1}) \times \prod_{i=1}^{n(p_d)} P(\boldsymbol{ku}_i \mid e_{start(i)-b}^{start(i)-1}, \boldsymbol{eu}_i, e_{end(i)+1}) \times \prod_{j=1}^{l(p_d)-1} P(z_j \mid z_{j-c-1}^{j-1}, e_{j-c}^{j+1}). \quad (2.15)$$

Equation 2.15 is the equation of our back transliteration method.

2.4 Beam search solution for context sensitive grammar

Equation 2.15 includes context-sensitive grammar. As such, it can not be carried out efficiently. In decoding from the head of a word to the tail, $e_{end(i)+1}$ in equation 2.15 becomes context-sensitive. Thus we try to get approximate results by using a beam search solution. To get the results, we use dynamic programming. Every node of ***eu*** in the lattice keeps the N-best results evaluated by using a letter of $e_{end(i)+1}$ that gives the maximum probability in the next letters. When the results of next node are evaluated for selecting the N-best, the accurate probabilities from the previous nodes are used.

2.5 Learning probability models based on the maximum entropy method

The probability models are learned based on the maximum entropy method. This makes it possible to prevent data sparseness relating to the model as well as to efficiently utilize many conditions, such as context, simultaneously. We use the Gaussian Prior (Chen and Rosenfeld, 1999) smoothing method for the language model. We use one Gaussian variance. We use the value of the Gaussian variance that minimizes the test set's perplexity.

The feature functions of the models based on the maximum entropy method are defined as combinations of letters. In addition, we use vowel, consonant, and semi-vowel classes for the translation model. We manually define the combinations of the letter positions such as e_j and e_{j-1}. The feature functions consist of the letter combinations that meet the combinations of the letter positions and are observed at least once in the learning data.

2.6 Corpus for learning

A Japanese-English word list aligned by unit was used for learning the translation model and the chunking model and for generating the lattice of conversion candidates. The alignment was done by semi-automatically. A romanized katakana character usually corresponds to one or several English letters or strings. For example, a romanized katakana character "*k*" usually corresponds to an English letter "c," "k," "ch," or "q." With such heuristic rules, the Japanese-English word corpus could be aligned by unit and the alignment errors were corrected manually.

3 Experiment

3.1 Learning data and test data

We conducted an experiment on back transliteration using English personal names. The learning data used in the experiment are described below.

The Dictionary of Western Names of 80,000 People[2] was used as the source of the Japanese-English word corpus. We chose the names in alphabet from A to Z and their corresponding katakana. The number of distinct words was 39,830 for English words and 39,562 for katakana words. The number of English-katakana pairs was 83,057[3]. We related the alphabet and katakana character units in those words by using the method described in section 2.6. We then used the corpus to make the translation and the chunking models and to generate a lattice of conversion candidates.

The learning of the language model was carried out using a word list that was created by merging two word lists: an American personal-

[2] Published by Nichigai Associates in Japan in 1994.

[3] This corpus includes many identical English-katakana word pairs.

name list[4], and English head words of the Dictionary of Western Names of 80,000 people. The American name list contains frequency information for each name; we also used the frequency data for the learning of the language model. A test set for evaluating the value of the Gaussian variance was created using the American name list. The list was split 9:1, and we used the larger data for learning and the smaller data for evaluating the parameter value.

The test data is as follows. The test data contained 333 katakana name words of American Cabinet officials, and other high-ranking officials, as well as high-ranking governmental officials of Canada, the United Kingdom, Australia, and New Zealand (listed in the World Yearbook 2002 published by Kyodo News in Japan). The English name words that were listed along with the corresponding katakana names were used as answer words. Words that included characters other than the letters A to Z were excluded from the test data. Family names and First names were not distinguished.

3.2 Experimental models

We used the following methods to test the individual effects of each factor of our method.

- Method A

Used a model that did not take English context into account. The plausibility is expressed as follows:

$$\hat{\boldsymbol{E}} = \arg\max_{\boldsymbol{E}} \prod_{i=1}^{n(p_d)} P(\boldsymbol{eu}_i \mid \boldsymbol{ku}_i) \,. \qquad (3.1)$$

- Method B

Used our language model and a translation model that did not consider English context. The constant $a = 3$ in the language model. The plausibility is expressed as follows:

$$\hat{\boldsymbol{E}} = \arg\max_{\boldsymbol{E}} \prod_{j=1}^{l(p_d)+1} P(e_j \mid e_{j-3}^{j-1}) \prod_{i=1}^{n(p_d)} P(\boldsymbol{ku}_i \mid \boldsymbol{eu}_i) \,. \qquad (3.2)$$

- Method C

Applied our chunking model to method B, with $c = 3$ in the chunking model. The plausibility is expressed as follows:

$$\hat{\boldsymbol{E}} = \arg\max_{\boldsymbol{E}} \prod_{j=1}^{l(p_d)+1} P(e_j \mid e_{j-3}^{j-1}) \prod_{i=1}^{n(p_d)} P(\boldsymbol{ku}_i \mid \boldsymbol{eu}_i) \times \prod_{j=1}^{l(p_d)-1} P(z_j \mid z_{j-4}^{j-1}, e_{j-3}^{j+4}). \qquad (3.3)$$

- Method D

Used our translation model that considered English context, but not the chunking model. $b = 3$ in the translation model. The plausibility is expressed as follows:

$$\hat{\boldsymbol{E}} = \arg\max_{\boldsymbol{E}} \prod_{j=1}^{l(p_d)+1} P(e_j \mid e_{j-3}^{j-1}) \times \prod_{i=1}^{n(p_d)} P\left(\boldsymbol{ku}_i \mid e_{start(i)-3}^{start(i)-1}, \boldsymbol{eu}_i, e_{end(i)+1}\right). \qquad (3.4)$$

- Method E

Used our language model, translation model, and chunking model. The plausibility is expressed as follows:

$$\hat{\boldsymbol{E}} = \arg\max_{\boldsymbol{E}} \prod_{j=1}^{l(p_d)+1} P(e_j \mid e_{j-3}^{j-1}) \times \prod_{i=1}^{n(p_d)} P\left(\boldsymbol{ku}_i \mid e_{start(i)-3}^{start(i)-1}, \boldsymbol{eu}_i, e_{end(i)+1}\right) \times \prod_{j=1}^{l(p_d)-1} P(z_j \mid z_{j-4}^{j-1}, e_{j-3}^{j+4}). \qquad (3.5)$$

3.3 Results

Table 1 shows the results of the experiment[5] on back transliteration from Japanese katakana to English. The conversion was determined to be successful if the generated English word agreed perfectly with the English word in the test data. Table 2 shows examples of back transliterated words.

Method	A	B	C	D	E
Top 1	23.7	57.4	61.6	63.1	66.4
Top 2	34.8	69.1	72.4	71.8	74.2
Top 3	42.9	73.6	76.6	75.4	79.3
Top 5	54.1	77.5	79.9	80.8	83.5
Top 10	63.4	82.0	85.3	86.5	87.7

Table 1: Ratio (%) of including the answer word in high-ranking words.

[4] Prepared from the 1990 Census conducted by the U.S. Department of Commerce. Available at http://www.census.gov/genealogy/names/ . The list includes 91,910 distinct words.

[5] For model D and E, we used N=50 for the beam search solution. In addition, we kept paths that represented parts of words existing in the learning data.

Japanese katakana (romanized katakana)	Created English
アシュクロフト (*a shu ku ro fu to*)	Ashcroft
キルシュシュタイン (*ki ru shu shu ta i n*)	Kirschstein
スペンサー (*su pe n sa -*)	Spencer
パウエル (*pa u e ru*)	Powell
プリンシピ (*pu ri n shi pi*)	Principi

Table 2: Example of English words produced.

4 Discussion

The correct-match ratio of the method E for the first-ranked words was 66%. Its correct-match ratio for words up to the 10th rank was 87%.

Regarding the top 1 ranked words, method B that used a language model increase the ratio 33-points from method A that did not use a language model. This demonstrates the effectiveness of the language model.

Also for the top 1 ranked words, method C which adopted the chunking model increase the ratio 4-points from method B that did not adopt the chunking model in the top 1 ranked words. This indicates the effectiveness of the chunking model.

Method D that used a translation model taking English context into account had a ratio 5-points higher in top 1 ranked words than that of method B that used a translation model not taking English context into account. This demonstrates the effectiveness of the language model.

Method E gave the best ratio. Its ratio for the top 1 ranked word was 42-points higher than method A's.

These results demonstrate the effectiveness of using English context for back transliteration.

5 Conclusion

This paper described a method for Japanese to English back transliteration. Unlike conventional methods, our method uses a target English context to calculate the plausibility of matching between English and katakana. Our method can treat English words that do not exist in learning data. We confirmed the effectiveness of our method in an experiment using personal names. We will apply this technique to cross-language information retrieval.

References

Hsin-Hsi Chen, Sheng-Jie Huang, Yung-Wei Ding, and Shih-Chung Tsai. 1998. *Proper Name Translation in Cross-Language Information Retrieval*. 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, pp.232-236.

Stanley F. Chen, Ronald Rosenfeld. 1999. *A Gaussian Prior for Smoothing Maximum Entropy Models*. Technical Report CMU-CS-99-108, Carnegie Mellon University.

Bonnie Glover Stalls and Kevin Knight. 1998. *Translating Names and Technical Terms in Arabic Text*. COLING/ACL Workshop on Computational Approaches to Semitic Languages.

Isao Goto, Naoto Kato, Noriyoshi Uratani, and Terumasa Ehara. 2003. *Transliteration Considering Context Information based on the Maximum Entropy Method*. Machine Translation Summit IX, pp.125-132.

Kil Soon Jeong, Sung Hyun Myaeng, Jae Sung Lee, and Key-Sun Choi. 1999. *Automatic Identification and Back-Transliteration of Foreign Words for Information Retrieval*. Information Processing and Management, Vol.35, No.4, pp.523-540.

Byung-Ju Kang and Key-Sun Choi. 2000. *Automatic Transliteration and Back-Transliteration by Decision Tree Learning*. International Conference on Language Resources and Evaluation. pp.1135-1411.

Kevin Knight and Jonathan Graehl. 1998. *Machine Transliteration*. Computational Linguistics, Vol.24, No.4, pp.599-612.

Wei-Hao Lin and Hsin-Hsi Chen. 2002. *Backward Machine Transliteration by Learning Phonetic Similarity*. 6th Conference on Natural Language Learning, pp.139-145.

Atsushi Fujii and Tetsuya Ishikawa. 2001. *Japanese/English Cross-Language Information Retrieval: Exploration of Query Translation and Transliteration*. Computers and the Humanities, Vol.35, No.4, pp.389-420.