# Transliteration Considering Context Information based on the Maximum Entropy Method

**Isao Goto, Naoto Kato, and Noriyoshi Uratani**

NHK Science and Technical Research Laboratories
1-11-10 Kinuta, Setagaya, Tokyo, Japan
{goto.i-es, katou.n-ga, uratani.n-fc }@nhk.or.jp

**Terumasa Ehara**

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

## Abstract

This paper proposes a method of automatic transliteration from English to Japanese words. Our method successfully transliterates an English word not registered in any bilingual or pronunciation dictionaries by converting each partial letters in the English word into Japanese *katakana* characters. In such transliteration, identical letters occurring in different English words must often be converted into different *katakana*. To produce an adequate transliteration, the proposed method considers chunking of alphabetic letters of an English word into conversion units and considers English and Japanese context information simultaneously to calculate the plausibility of conversion. We have confirmed experimentally that the proposed method improves the conversion accuracy by 63% compared to a simple method that ignores the plausibility of chunking and contextual information.

## 1 Introduction

During the translation of different character sets from English to Japanese, technical terms and proper nouns are generally replaced with approximate phonetic equivalents. Such translation is called transliteration. Because technical terms and proper nouns are not always listed in bilingual dictionaries, these technical terms and proper nouns, especially new ones, are normally difficult to translate. The ability to transliterate automatically without reference to a dictionary would make it possible to automate the generation of Japanese words from unknown English words. Automatic transliteration is particularly useful in cross-language information retrieval and machine translation.

This paper presents a transliteration method that can generate a Japanese *katakana* word from an unknown English word that is not registered in any bilingual or pronunciation dictionaries. Treating such unknown words, an English word is divided into conversion units that are partial English character strings in an English word and each English conversion unit is converted into a partial *katakana* character string.

To produce an adequate transliteration, our method applies the following three approaches: It calculates the likelihood of a particular choice of letter chunking into English conversion units for an English word. It considers the English and Japanese contextual information simultaneously to calculate the plausibility of conversion from each English conversion unit to various Japanese conversion candidate units using a single probability model. And it uses probability models based on the maximum entropy method that can treat different kind information.

## 2 Conversion units based English to Japanese transliteration

In the case of transliteration to Japanese, special characters called *katakana* are used to indicate how a word is pronounced. For example, a transliteration into *katakana* of the word "actinium" is written "アクチニウム(*a ku chi ni u*

*mu*)." (Here, the italics indicate Romanized *katakana* characters.) Handling an unknown word that does not registered in any dictionaries requires that the word must be divided into certain conversion units. Figure 1 shows an example of conversion based English to Japanese transliteration.

English: actinium

⇩

English conversion units: a/c/ti/ni/u/m

⇩

Converted Japanese units: ア/ク/チ/ニ/ウ/ム
(*a*/*ku*/*chi*/*ni*/*u*/*mu*)

⇩

Japanese: アクチニウム
(*a ku chi ni u mu*)

Figure 1: Example of transliteration

Transliteration is a process of generating a translated word by expressing the pronunciation of the original foreign word using characters of the destination language. Thus, the central problem in transliteration is predicting the pronunciation of the original word.

Transliteration is a difficult task, since determining the precise pronunciation of an English word can be quite difficult. For example, when occurring in an English word, the character "u" can be pronounced "*a*," or "*u*," or in some other fashion. (Here, the italic characters represent Romanized *katakana*.) The context of the "u" determines the correct pronunciation. In the word "actinium," "u" is pronounced "*u*," while "u" in "inductance" is pronounced "*a*." The pronunciation of each character in a word changes with the surrounding characters – that is, with the context. Thus, contextual information surrounding the characters in question, is essential in determining pronunciation of target characters in an English word.

A single phoneme may occasionally consist of more than a single character; for example, the "ou" in "doublet" is pronounced "*a*," but is pronounced "*o*-" in "resource." In such cases, pronunciation is determined by the unit "ou" and its context.

Phonetic conversion units that vary in length pose the problem of where such units are to be delimited. For example, the word "doublet" may be divided as d/o/u/b/l/e/t, d/o/u/b/le/t, d/ou/b/l/e/t, dou/b/le/t, or dou/b/l/e/t. A successful transliteration system must calculate the respective probabilities of chunking into these various conversion units.

We will discuss the initial implementation of our transliteration technique in subsection 2.1 and 2.2, then go on to discuss its learning phases in subsections 2.3 to 2.6.

## 2.1 Lattice of conversion candidate units and the corresponding notation

An English word, $\boldsymbol{E}$, is expressed by Equation 2.1, with "^" and "$" added as prefix and suffix., respectively.

$$\boldsymbol{E} = e_0^{m+1} = e_0 e_1 ... e_{m+1} \quad (2.1)$$

$$e_0 = \text{^},\ e_{m+1} = \$ \quad (2.2)$$

where $e_j$ is the $j$-th character in the English word and $m$ is the number of characters, except for "^" and "$" and $e_0^{m+1}$ is a character string from $e_0$ to $e_{m+1}$. A lattice of *katakana* candidates $\mathcal{L}\{\boldsymbol{K}\}$ is generated by a set of rules that generate conversion candidate units, made up of the English units ***eu***

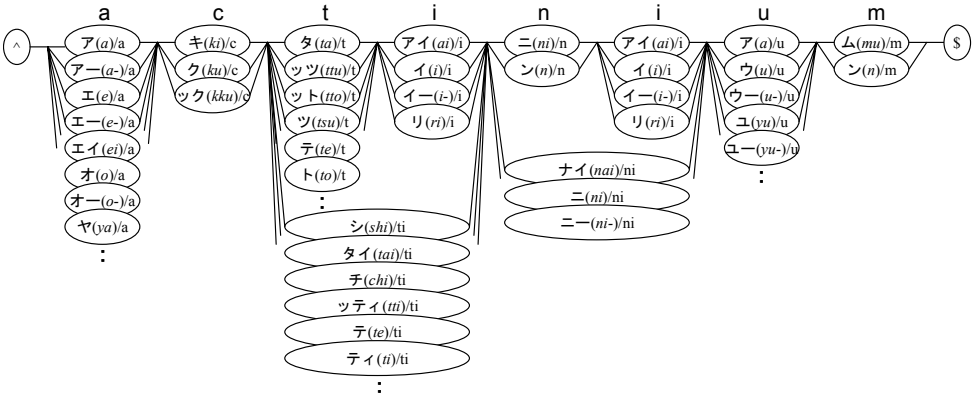


Figure 2: Example of $\mathcal{L}\{\boldsymbol{K}\}$ of "actinium" (***ku*/*eu***)

and corresponding *katakana* units $\boldsymbol{ku}$. $\mathcal{L}\{\boldsymbol{K}\}$ is made up of all $\boldsymbol{ku}$ corresponding to $\boldsymbol{eu}$ by using every $\boldsymbol{eu}$ of conversion candidate unit generating rules applicable to the characters in $\boldsymbol{E}$. An example of $\mathcal{L}\{\boldsymbol{K}\}$ for "actinium" is shown in figure 2. Each node in figure 2 represents $\boldsymbol{ku}$/$\boldsymbol{eu}$.

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

Next, a certain path, $p_d$, in $\mathcal{L}\{\boldsymbol{K}\}$ is selected. The number of character units other than "^" and "$" in $p_d$ is expressed by $n(p_d)$. The character units in $p_d$ are numbered from beginning to end. Then, the *katakana* word, $\boldsymbol{K}$, resulting from the conversion of an English word, $\boldsymbol{E}$, for $p_d$ is expressed as follows:

$$\boldsymbol{E} = e_0^{m+1} = e_0 e_1 ... e_{m+1} = \boldsymbol{eu}_0^{n(p_d)+1} = \boldsymbol{eu}_0 \boldsymbol{eu}_1 .. \boldsymbol{eu}_{n(p_d)+1} \quad (2.3)$$

$$\boldsymbol{K} = k_0^{l(p_d)+1} = k_0 k_1 ... k_{l(p_d)+1}$$

$$= \boldsymbol{ku}_0^{n(p_d)+1} = \boldsymbol{ku}_0 \boldsymbol{ku}_1 .. \boldsymbol{ku}_{n(p_d)+1} \quad (2.4)$$

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

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

where $k_j$ is the $j$-th character in the *katakana* word. Meanwhile, $l(p_d)$ is the number of characters other than "^" and "$" in the *katakana* word. $\boldsymbol{eu}_0^{n(p_d)+1}$ in Equation 2.3 shows the units in which the characters are grouped in the English word for use in providing conversion candidates. $\boldsymbol{ku}_0^{n(p_d)+1}$ in Equation 2.4 for each $p_d$ in $\mathcal{L}\{\boldsymbol{K}\}$ provides the converted *katakana* word candidates.

## 2.2 Determining the plausibility considering context information

To determine the corresponding *katakana* word for an English word, we must solve for $\boldsymbol{K}$ in the following equation:

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

$\hat{\boldsymbol{K}}$ means a estimated *katakana* word. However, it is quite difficult to solve Equation 2.6 directly for an unknown word. Thus, the word in Equation 2.6 is broken down into character units as shown in Equation 2.7, using Equations 2.3 and 2.4.

$$\hat{\boldsymbol{K}} = \arg\max_{\boldsymbol{K}} \sum_{\boldsymbol{eu}_0^{n(p_d)+1}} \sum_{\boldsymbol{ku}_0^{n(p_d)+1}} \left\{ P(\boldsymbol{K} \mid \boldsymbol{ku}_0^{n(p_d)+1}) \times P(\boldsymbol{ku}_0^{n(p_d)+1} \mid \boldsymbol{eu}_0^{n(p_d)+1}) P(\boldsymbol{eu}_0^{n(p_d)+1} \mid \boldsymbol{E}) \right\} \quad (2.7)$$

Equation 2.7 indicates that a result is rendered by the sum of probabilities of all units of the same word.

The model $P(\boldsymbol{K} \mid \boldsymbol{ku}_0^{n(p_d)+1})$ in equation 2.7 is the probability distribution that characters of conversion candidate units of *katakana* $\boldsymbol{ku}_0^{n(p_d)+1}$ correctly correspond to the characters of a word in *katakana* $\boldsymbol{K}$. The model $P(\boldsymbol{ku}_0^{n(p_d)+1} \mid \boldsymbol{eu}_0^{n(p_d)+1})$ is the probability distribution of corresponding conversion candidate units of *katakana* $\boldsymbol{ku}_0^{n(p_d)+1}$ to conversion units of English $\boldsymbol{eu}_0^{n(p_d)+1}$. We call it the translation model. The model $P(\boldsymbol{eu}_0^{n(p_d)+1} \mid \boldsymbol{E})$ is the probability distribution for chunking of $\boldsymbol{E} = e_0^{m+1}$ into conversion units $\boldsymbol{eu}_0^{n(p_d)+1}$. We call it the chunking model.

Here, we show an instance of $P(\boldsymbol{K} \mid \boldsymbol{ku}_0^{n(p_d)+1}) P(\boldsymbol{ku}_0^{n(p_d)+1} \mid \boldsymbol{eu}_0^{n(p_d)+1}) P(\boldsymbol{eu}_0^{n(p_d)+1} \mid \boldsymbol{E})$ with a concrete value as follows:

$$P(\text{^アクチニウム\$} \mid \underset{(a\ ku\ chi\ ni\ u\ mu)}{\text{^}} / \underset{(a\ /\ ku\ /\ chi\ /\ ni\ /\ u\ /\ mu)}{\text{ア/ク/チ/ニ/ウ/ム}} / \$) \times$$

$$P(\text{^}/\underset{(a\ /\ ku\ /\ chi\ /\ ni\ /\ u\ /\ mu)}{\text{ア/ク/チ/ニ/ウ/ム}} / \$ \mid \text{^}/\text{a/c/ti/ni/u/m}/\$) \times$$

$$P(\text{^}/\text{a/c/ti/ni/u/m}/\$ \mid \text{^actinium\$})$$

$P(\boldsymbol{ku}_0^{n(p_d)+1} \mid \boldsymbol{eu}_0^{n(p_d)+1})$ can be transformed from processing by a word to processing by a conversion units.

$$P(\boldsymbol{ku}_0^{n(p_d)+1} \mid \boldsymbol{eu}_0^{n(p_d)+1}) = \prod_{i=1}^{n(p_d)+1} P(\boldsymbol{ku}_i \mid \boldsymbol{ku}_0^{i-1}, \boldsymbol{eu}_0^{n(p_d)+1}) \quad (2.8)$$

We reduce the condition of $P(\boldsymbol{ku}_i \mid \boldsymbol{ku}_0^{i-1}, \boldsymbol{eu}_0^{n(p_d)+1})$. The conditional information of English is to be $\boldsymbol{eu}_i$ and as many as $a$ characters and $b$ characters before and after $\boldsymbol{eu}_i$, respectively. The conditional information of *katakana* is to be $c$ characters before $\boldsymbol{ku}_i$.

$$\prod_{i=1}^{n(p_d)+1} P(\boldsymbol{ku}_i \mid \boldsymbol{ku}_0^{i-1}, \boldsymbol{eu}_0^{n(p_d)+1})$$

$$\approx \prod_{i=1}^{n(p_d)+1} P(\boldsymbol{ku}_i \mid k_{start\_ku(i)-c}^{start\_ku(i)-1}, e_{start\_eu(i)-a}^{start\_eu(i)-1}, \boldsymbol{eu}_i, e_{end\_eu(i)+1}^{end\_eu(i)+b}) \quad (2.9)$$

where $start\_eu(i)$ is the top position of the $i$-th character unit $\boldsymbol{eu}_i$, while $end\_eu(i)$ is the last

position of the $i$-th character unit $\boldsymbol{eu}_i$, $start\_ku(i)$ is the top position of the $i$-th character unit $\boldsymbol{ku}_i$; and $a$, $b$, and $c$ are constant.

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

$$z_j = \begin{cases} \text{delimiter} \\ \text{not delimiter} \end{cases} \tag{2.10}$$

The chunking model is transformed into processing by a character using $z_j$.

$$P(\boldsymbol{eu}_0^{n(p_d)+1} \mid \boldsymbol{E}) = \prod_{j=1}^{m-1} P(z_j \mid z_0^{j-1}, e_0^{m+1}) \tag{2.11}$$

We reduce the condition of $P(z_j \mid z_0^{j-1}, e_0^{m+1})$. The conditional information of English is to be as many as $a'$ characters and $b'$ characters before and after $z_j$, respectively. And the conditional information of $z_0^{j-1}$ is to be $c'$ $z$ before $z_j$.

$$\prod_{j=1}^{m-1} P(z_j \mid z_0^{j-1}, e_0^{m+1}) \approx \prod_{j=1}^{m-1} P(z_j \mid z_{j-c'}^{j-1}, e_{j-a'+1}^{j+b'}) \tag{2.12}$$

Equation 2.7 is transformed using equation 2.9 and 2.12.

$$\hat{\boldsymbol{K}} \approx \arg\max_{\boldsymbol{K}} \sum_{\boldsymbol{eu}_0^{n(p_d)+1}} \sum_{\boldsymbol{ku}_0^{n(p_d)+1}} \left\{ P(\boldsymbol{K} \mid \boldsymbol{ku}_0^{n(p_d)+1}) \times \right.$$
$$\prod_{i=1}^{n(p_d)+1} P(\boldsymbol{ku}_i \mid k_{start\_ku(i)-c}^{start\_ku(i)-1}, e_{start\_eu(i)-a}^{start\_eu(i)-1}, \boldsymbol{eu}_i, e_{end\_eu(i)+1}^{end\_eu(i)+b}) \times$$
$$\left. \prod_{j=1}^{m-1} P(z_j \mid z_{j-c'}^{j-1}, e_{j-a'+1}^{j+b'}) \right\} \tag{2.13}$$

Equation 2.13 is our proposed plausibility.

And we more approximate Equation 2.13 to selecting one $\boldsymbol{ku}_0^{n(p_d)+1}$ and $\boldsymbol{eu}_0^{n(p_d)+1}$.

$$\hat{\boldsymbol{K}} \approx \arg\max_{\boldsymbol{K} \approx \boldsymbol{ku}_0^{n(p_d)+1}} \prod_{i=1}^{n(p_d)+1} P\left(\boldsymbol{ku}_i \mid k_{start\_ku(i)-c}^{start\_ku(i)-1}, e_{start\_eu(i)-a}^{start\_eu(i)-1}, \right.$$
$$\left. \boldsymbol{eu}_i, e_{end\_eu(i)+1}^{end\_eu(i)+b} \right) \prod_{j=1}^{m-1} P(z_j \mid z_{j-c'}^{j-1}, e_{j-a'+1}^{j+b'}) \tag{2.14}$$

where one $\boldsymbol{ku}_0^{n(p_d)+1}$ is regarded as $\boldsymbol{K}$.

Equation 2.14 is our proposed alternate plausibility. Equation 2.14 can be naively carried out efficiently based on Viterbi algorithm.

We devised the probability models based on the maximum entropy method.

### 2.3 Preparation of corpus

In this section, we describe the method of creating a corpus that links the character units in each English word to the correct *katakana* equivalent. We semi automatically prepared a corpus using a Japanese *katakana*-English dictionary. The following is a brief explanation of this semi automatic character-unit-relating process applied (for example) to the English word "thesaurus," and corresponding *katakana* word "シソーラス(*shi so – ra su*)."[1]

1) Romanize the *katakana*.
[thesaurus:*shi so – ra su*]
2) Use a manually made list[2] relating Roman character units with English character units to identify character matching between English characters and Roman characters.[3] At the same time, use the information on the positions of individual characters and their orders in the word. Figure 3 shows an example illustrating a match between character units and the optimal path. The remaining units that have corresponding units are then matched. If no corresponding character unit exists, the units are merged with preceding units.
[th/e/s/au/r/u/s:*sh/i/s/o-/r/a/s/u*]
3) Convert the Romanized syllables back to *katakana* characters. If a delimiter occurs in a *katakana* phoneme, remove it. Remove the corresponding delimiters in the English word.
[the/sau/ru/s:シ(*shi*)/ソー(*so-*)/ラ(*ra*)/ス(*su*)]
4) Reconfigure the delimited positions and add delimiters using a post filter. In the last step, manually correct errors and complete the corpus.
[the/sau/ru/s:シ(*shi*)/ソー(*so-*)/ラ(*ra*)/ス(*su*)]

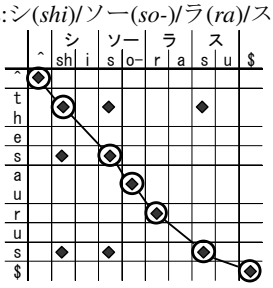


Figure 3: Example of points of correspondence between character units and the optimal path

[1] The *katakana* character "ー(-)" lengthens vowel.

[2] We use relating rules as many as 153 and exception rule for x in English.

[3] The character "ッ (small *tsu*)" that lengthens consonant is merged with the subsequent characters and it is not to be the object for matching.

The results of character-unit-relating based on our method don't have NULL character unit.

Because the English character units are determined by relating them to *katakana* characters expressing syllables, the English character units become units that consider phonetic aspects.

### 2.4 Preparation of rules for generating conversion candidate units

Using the corpus in which character-unit-relating has been completed, we prepare rules for character unit conversion from English character units into *katakana* character units. For example, conversion rules such as "the"->"シ(*shi*)", "sau"->"ソー(*so-*)", "ru"->"ラ(*ra*)", and "s"->"ス(*su*)" can be derived from [the/sau/ru/s: シ(*shi*)/ ソー(*so-*)/ ラ(*ra*)/ ス(*su*)].

### 2.5 Feature functions used in the maximum entropy method

In Equation 2.13 and 2.14, we define feature functions of a translation model and a chunking model for learning based on the maximum entropy method. One important consideration in devising a model based on the maximum entropy method is the question of which factors to include among the feature functions. Positing a short distance to the character unit of interest and continuity as important factors, we determined the conditions of the feature functions by combining attributes.

For translation model $P(\boldsymbol{ku}_i \mid k_{start\_ku(i)-c}^{start\_ku(i)-1}, e_{start\_eu(i)-a}^{start\_eu(i)-1}, \boldsymbol{eu}_i, e_{end\_eu(i)+1}^{end\_eu(i)+b})$, we used information on phonetic types of consonants, vowels, and semi-vowels, in addition to character information of English, in order to prevent data sparseness. Information on the consonants, vowels, and semi-vowels of $e$ is expressed by $G(e)$.

$$G(e) = \begin{cases} \text{vowel} & (e \in \{\text{a,i,u,e,o}\}) \\ \text{semi-vowel} & (e \in \{\text{h,y}\}) \\ \text{consonant} & (\text{else}) \end{cases} \quad (2.15)$$

$\boldsymbol{eu}_i$, $e$, $G(e)$, $k$, and $\boldsymbol{ku}_i$ are handled as independent attributes. The combinations for feature functions are follows:

- $\boldsymbol{ku}_i$ and $\boldsymbol{eu}_i$
- $\boldsymbol{ku}_i$, $\boldsymbol{eu}_i$, and some $e$ before or after or around $\boldsymbol{eu}_i$
- $\boldsymbol{ku}_i$, $\boldsymbol{eu}_i$, and some $G(e)$ before or after or around $\boldsymbol{eu}_i$
- $\boldsymbol{ku}_i$ and $k_{start\_ku(i)-1}$

For chunking model $P(z_j \mid z_{j-c'}^{j-1}, e_{j-a'+1}^{j+b'})$, the combination for feature functions are follows:

- $z_j$, $e_j$, and $e_{j+1}$
- $z_j$, $e_j$, $e_{j+1}$, and some $z$ and $e$ before $e_j$
- $z_j$, $e_j$, $e_{j+1}$, and some $e$ after $e_j$
- $z_j$, $e_j$, $e_{j+1}$, and some $z$ and $e$ before $e_j$ and some $e$ around $e_j$

## 3 Experiment

An overview of the experiment is given below. We collect English-*katakana* corresponding pairs in a Japanese-English dictionary, then divide them into learning data and test data. We create a corpus from the learning data, then prepare conversion rules and perform model learning using the corpus. We then carry out transliteration using the test data.

The data used in the experiment consists of as many as 15,135 English-*katakana* pairs selected from a Japanese *katakana* - English Dictionary[4]. Words of origins other than English and those including capital letters were excluded from such pairs. When a headword was made up of two or more words, English-*katakana* pairs were made for the individual words. That is, for compound headwords, we created two pairs. Redundant pairs were removed.

### 3.1 Features of the data

For the test data, we selected one thousand English words, each consisting of four or more characters,[5] from the entire data set. The remaining data and its corresponding *katakana* words was the learning data. The learning data did not include English words in the test data.

In the test data, for cases in which two or more corresponding *katakana* words were found for an English word, all were regarded as correct. The average number of corresponding *katakana* words for each English word in the test data was 1.07. The average length of the English words in the test data in number of characters was 7.83.

---

[4] Sanseido's Concise Dictionary of Katakana words (second edition)

[5] Words of three or fewer characters are primarily basic words, and their types are limited. The words having at least four characters were used in the experiment, as they comprise the majority of what may require transliteration.

For the learning data, the results of relating the character units in each word in the learning data are as follows. For English words, the average number of characters in each character unit was 1.80; for *katakana* words, the figure was 1.31.

### 3.2 Probability model learning

The learning based on the maximum entropy method was based on conditions of feature functions observed at least once in the learning data among the combinations of attributes. Parameters are estimated by Berger's method (1996). The estimation was repeated 500 times.

### 3.3 Experimental models

To clarify the contribution of each factor, we created the following models. The parameters of the chunking model are $a' = b' = c' = 4$ .

- Model A

This translation model does not consider context information.

$$\hat{\boldsymbol{K}} \approx \arg\max_{\boldsymbol{K}} \sum_{\boldsymbol{eu}_0^{n(p_d)+1}} \sum_{\boldsymbol{ku}_0^{n(p_d)+1}} \left\{ P(\boldsymbol{K} \mid \boldsymbol{ku}_0^{n(p_d)+1}) \times \prod_{i=1}^{n(p_d)+1} P(\boldsymbol{ku}_i \mid \boldsymbol{eu}_i) \prod_{j=1}^{m-1} P(z_j \mid z_{j-4}^{j-1}, e_{j-3}^{j+4}) \right\} \quad (3.1)$$

- Model B

Using Bayes theorem, a Japanese language model is introduced.

$$\hat{\boldsymbol{K}} \approx \arg\max_{\boldsymbol{K}} \sum_{\boldsymbol{eu}_0^{n(p_d)+1}} \sum_{\boldsymbol{ku}_0^{n(p_d)+1}} \left\{ P(\boldsymbol{K} \mid \boldsymbol{ku}_0^{n(p_d)+1}) \times \prod_{i=1}^{n(p_d)+1} P(\boldsymbol{ku}_i \mid k_{start\_ku(i)-1}) P(\boldsymbol{eu}_i \mid \boldsymbol{ku}_i) \prod_{j=1}^{m-1} P(z_j \mid z_{j-4}^{j-1}, e_{j-3}^{j+4}) \right\} \quad (3.2)$$

- Model C

Approximating a translation model into two independent models, a Japanese language model is introduced.

$$\hat{\boldsymbol{K}} \approx \arg\max_{\boldsymbol{K}} \sum_{\boldsymbol{eu}_0^{n(p_d)+1}} \sum_{\boldsymbol{ku}_0^{n(p_d)+1}} \left\{ P(\boldsymbol{K} \mid \boldsymbol{ku}_0^{n(p_d)+1}) \times \left( \prod_{i=1}^{n(p_d)+1} P(\boldsymbol{ku}_i \mid k_{start\_ku(i)-1}) \right)^{1-\alpha} \left( \prod_{i=1}^{n(p_d)+1} P(\boldsymbol{ku}_i \mid \boldsymbol{eu}_i) \right)^{\alpha} \times \prod_{j=1}^{m-1} P(z_j \mid z_{j-4}^{j-1}, e_{j-3}^{j+4}) \right\} \quad (3.3)$$

- Model D

The translation model considers Japanese contextual information.

$$\hat{\boldsymbol{K}} \approx \arg\max_{\boldsymbol{K}} \sum_{\boldsymbol{eu}_0^{n(p_d)+1}} \sum_{\boldsymbol{ku}_0^{n(p_d)+1}} \left\{ P(\boldsymbol{K} \mid \boldsymbol{ku}_0^{n(p_d)+1}) \times \prod_{i=1}^{n(p_d)+1} P(\boldsymbol{ku}_i \mid k_{start\_ku(i)-1}, \boldsymbol{eu}_i) \prod_{j=1}^{m-1} P(z_j \mid z_{j-4}^{j-1}, e_{j-3}^{j+4}) \right\} \quad (3.4)$$

- Model E

The translation model considers English context information. The parameters are $a = b = 3, c = 0, a' = b' = c' = 4$ in the method proposed by equation 2.13.

$$\hat{\boldsymbol{K}} \approx \arg\max_{\boldsymbol{K}} \sum_{\boldsymbol{eu}_0^{n(p_d)+1}} \sum_{\boldsymbol{ku}_0^{n(p_d)+1}} \left\{ P(\boldsymbol{K} \mid \boldsymbol{ku}_0^{n(p_d)+1}) \times \prod_{i=1}^{n(p_d)+1} P\left(\boldsymbol{ku}_i \mid e_{start\_eu(i)-3}^{start\_eu(i)-1}, \boldsymbol{eu}_i, e_{end\_eu(i)+1}^{end\_eu(i)+3}\right) \times \prod_{j=1}^{m-1} P(z_j \mid z_{j-4}^{j-1}, e_{j-3}^{j+4}) \right\} \quad (3.5)$$

- Model F

Approximating a translation model into two models, a Japanese language model is introduced. The translation model also considers English context information.

$$\hat{\boldsymbol{K}} \approx \arg\max_{\boldsymbol{K}} \sum_{\boldsymbol{eu}_0^{n(p_d)+1}} \sum_{\boldsymbol{ku}_0^{n(p_d)+1}} \left\{ P(\boldsymbol{K} \mid \boldsymbol{ku}_0^{n(p_d)+1}) \times \left( \prod_{i=1}^{n(p_d)+1} P(\boldsymbol{ku}_i \mid k_{start\_ku(i)-1}) \right)^{1-\alpha} \times \left( \prod_{i=1}^{n(p_d)+1} P\left(\boldsymbol{ku}_i \mid e_{start\_eu(i)-3}^{start\_eu(i)-1}, \boldsymbol{eu}_i, e_{end\_eu(i)+1}^{end\_eu(i)+3}\right) \right)^{\alpha} \prod_{j=1}^{m-1} P(z_j \mid z_{j-4}^{j-1}, e_{j-3}^{j+4}) \right. \quad (3.6)$$

- Model G

The translation model considers both English and Japanese contextual information. The parameters are $a = b = 3$ , $c = 1$ , $a' = b' = c' = 4$ in the method proposed by equation 2.13.

$$\hat{\boldsymbol{K}} \approx \arg\max_{\boldsymbol{K}} \sum_{\boldsymbol{eu}_0^{n(p_d)+1}} \sum_{\boldsymbol{ku}_0^{n(p_d)+1}} \left\{ P(\boldsymbol{K} \mid \boldsymbol{ku}_0^{n(p_d)+1}) \times \prod_{i=1}^{n(p_d)+1} P\left(\boldsymbol{ku}_i \mid k_{start\_ku(i)-1}, e_{start\_eu(i)-3}^{start\_eu(i)-1}, \boldsymbol{eu}_i, e_{end\_eu(i)+1}^{end\_eu(i)+3}\right) \times \prod_{j=1}^{m-1} P(z_j \mid z_{j-4}^{j-1}, e_{j-3}^{j+4}) \right\} \quad (3.7)$$

- Model H

The translation model considers both English and Japanese contextual information. The parameters are $a = b = 3$ , $c = 1$ , $a' = b' = c' = 4$ in the method proposed by equation 2.14.

$$\hat{\boldsymbol{K}} \approx \arg\max_{\boldsymbol{K} \approx \boldsymbol{ku}_o^{n(p_d)+1}} \prod_{i=1}^{n(p_d)+1} P\left(\boldsymbol{ku}_i \mid k_{start\_ku(i)-1}, e_{start\_eu(i)-3}^{start\_eu(i)-1}, \boldsymbol{eu}_i, e_{end\_eu(i)+1}^{end\_eu(i)+3}\right) \prod_{j=1}^{m-1} P(z_j \mid z_{j-4}^{j-1}, e_{j-3}^{j+4}) \quad (3.8)$$

### 3.4 Experimental results

Table 1 shows the results[6] of transliteration from English to *katakana*. The word agreement ratio (W.A.) and the character agreement ratio (C.A.) are shown for all models and the case that chunking model $P(\boldsymbol{eu}_0^{n(p_d)+1} \mid \boldsymbol{E})$ is approximated into constant, namely without chunking model, for all models.

The character agreement ratio is defined by:

$$\text{C.A.} = \frac{L - Err}{L} \tag{3.9}$$

where $L$ is the number of characters of the correct word and $Err$ is the number of error characters of the generated word. Calculation results smaller than zero are treated as zero. Here, the number of error characters was counting the minimum number of operations of 1) character insertion, 2) character deletion, and 3) character replacement for correction of the generated word to provide the correct word.

Table 1: Experimental results(%)

| | | Chunking model | | | |
|---|---|---|---|---|---|
| | | Use | | Not use | |
| Model | | W.A. | C.A. | W.A. | C.A. |
| A | | 35.4 | 77.6 | 7.1 | 42.2 |
| B | | 45.7 | 80.5 | 30.1 | 72.7 |
| C | $\alpha$=0.5 | 45.8 | 80.6 | 29.9 | 72.6 |
| | $\alpha$=0.9 | 40.1 | 79.1 | 23.4 | 67.0 |
| D | | 46.7 | 80.0 | 11.1 | 47.2 |
| E | | 70.2 | 91.0 | 63.6 | 87.6 |
| F | $\alpha$=0.5 | 65.5 | 88.8 | 53.5 | 85.2 |
| | $\alpha$=0.9 | 66.2 | 89.3 | 62.3 | 87.5 |
| G | | 70.5 | 91.0 | 63.5 | 87.4 |
| H | | 70.5 | 91.0 | 63.5 | 87.4 |

### 3.5 Discussion

- Effects of considering a Japanese context

Of the translation models, models B, C, and D consider the Japanese context but not the English context. These models provide similar conversion results, with W.A. values of 46-47%, about 10% higher than model A, in which the Japanese context is not considered. This demonstrates the effectiveness of considering the Japanese context.

- Effects of considering an English context

Of the translation models, the W.A. for model E, which considered English context information, was 35% higher than for model A, which did not. This indicates the effectiveness of considering English context information in translation models. Predictions that consider the English context rather than the Japanese context provided greater accuracy.

- Efficiency of combining a language model and a translation model into a single model

One probability model gives a W.A. of about 47% for a model D that considers the Japanese context and incorporates an English conversion unit. This is somewhat better than the W.A. for models B and C, which have distinct language and translation models.

Consider the translation models that consider the English context. Compare the results of model E, which does not consider the Japanese context, and those of models F and G, which do. For model F, which considers the Japanese context, a language model is introduced distinct from a translation model. The resulting W.A. is lower than in model E. For $\alpha$ = 0.5, W.A. is 5% lower. For $\alpha$ = 0.9, W.A. is 4% lower, in which case, the effect of the language model is smaller than $\alpha$ = 0.5. Introducing a language model distinct from a translation model reduces overall accuracy. For model G, which simultaneously considers both Japanese and English contexts using a single probabilistic model, some improvement in W.A. was noted. These results indicate that our method can achieve greater efficiency by considering all information.

- Effects of a chunking model

A chunking model produces a higher W.A. for all models, compared to corresponding models without the chunking model. This indicates the effectiveness of our chunking model.

- Effects of using $G(e)$ in translation models

The W.A. for model G without $G(e)$ was 69.2%. So the W.A. is improved about 1% by using $G(e)$. This demonstrates that the information provided by $G(e)$ is of some benefit.

- Same solution by different paths in $\mathcal{L}\{\boldsymbol{K}\}$

Almost no difference can be seen between the result of models G and H. There are two reasons.

[6] Equation 3.1-3.7 can not be carried out efficiently. Thus, we use lattice paths ranked in the top 10 of the probabilities. It is possible to obtain high-ranking lattice paths efficiently based on dynamic programming.

In the lattice of conversion candidates, relatively few paths will produce the same word. In the paths of correct answer in $\mathcal{L}\{\boldsymbol{K}\}$, the average number of the paths is 1.08. Another reason is that the large differences in probabilities of the paths. In the paths in $\mathcal{L}\{\boldsymbol{K}\}$ for model G and H, the probability values of the first path divided by that of the tenth path is larger than ten in 98.8% of the cases.

The W.A. for the proposed models G and H is 63% greater than when using no contextual information (model A) and dispensing chunking model. This demonstrates the effectiveness of our methods embodied in Equations 2.13 and 2.14.

## 4 Related works

Some other transliteration methods have been proposed. I.Kang and Kim (2000) use a character unit as the conversion unit in consideration the phonetic aspects of transliteration from English to Korean. However, their method does not use context information, which plays an important role in our method.

Oh and Choi (2002) has proposed an English to Korean transliteration method that determines how to chunk English characters into conversion units based on decision trees. However, this method determines one set of conversion units before considering the information for the Korean conversion units that correspond to English conversion units. In contrast, our method simultaneously calculates the probabilities for conversion candidate units corresponding to the original units and the chunking probabilities for conversion units. Additionally, their method does not use context information to calculate probabilities when predicting English pronunciation.

There is an English to Korean method considers context information by using decision trees to select the Korean character units corresponding to each English letter (B.Kang and Choi, 2000). The conversion accuracy of their method is lower than methods that use conversion units, taking into consideration the phonetic aspects of the original English source (I.Kang and Kim, 2000).

Methods (Jung et al., 2000; Oh and Choi, 2002) have been proposed that simply looks up the English pronunciation in pronunciation dictionaries for transliterating English to Korean. However, methods that assume the English pronunciation is already known cannot convert a new word for which no pronunciation is listed in a dictionary.

Knight and Graehl (1998) have undertaken research related to back transliteration, by which the original English word is determined from a transliterated Japanese word. Their English letter-to-sound WSFT treats English letters in word units based on an English pronunciation dictionary. Thus, if the method is applied to transliteration, it is unable to handle English words not registered in a pronunciation dictionary.

## 5 Conclusion

This paper describes an English-to-Japanese transliteration method which consisted of considering English and Japanese contexts simultaneously by a single probability model; and calculating the plausibility of chunking for English letters into English conversion units. Our proposed method can make effective use of context information. Our experimental results indicate the effectiveness of our approaches. We will refine the conversion accuracy of this method by positing the origin of each word expected to provide useful information for accurate transliteration.

## 6 Bibliographical References


Berger, Adam L., Stephen A. Della Pietra, and Vincent J. Della Pietra. (1996). *A Maximum Entropy Approach to Natural Language Processing*. CL, Vol.22, No.1, pp.39-71.

Jung, Sung Young, Sung Lim Hong, and Eunok Paek. (2000). *An English to Korean Transliteration Model of Extended Markov Window*. COLING, Vol.1, pp.383-389.

Kang, Byung-Ju and Key-Sun Choi. (2000). *Automatic Transliteration and Back-Transliteration by Decision Tree Learning*. LREC, pp.1135-1411.

Kang, In-Ho and GilChang Kim. (2000). *English-to-Korean Transliteration using Multiple Unbounded Overlapping Phoneme Chunks*. COLING, Vol.1, pp.418-424.

Knight, Kevin and Jonathan Graehl. (1998). *Machine Transliteration*. CL, Vol.24, No.4, pp.599-612.

Oh, Jong-Hoon and Key-Sun Choi. (2002). *An English-Korean Transliteration Model Using Pronunciation and Contextual Rules*. COLING.