Journal of Artificial Intelligence Research 27 (2006) 119–151 Submitted 1/06; published 10/06

A Comparison of Different Machine Transliteration Models

Jong-Hoon Oh ROVELLIA@NICT.GO.JP
Computational Linguistics Group
National Institute of Information and Communications Technology (NICT)
3-5 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0289 Japan

Key-Sun Choi KSCHOI@CS.KAIST.AC.KR
Computer Science Division, Department of EECS
Korea Advanced Institute of Science and Technology (KAIST)
373-1 Guseong-dong, Yuseong-gu, Daejeon 305-701 Republic of Korea

Hitoshi Isahara ISAHARA@NICT.GO.JP
Computational Linguistics Group
National Institute of Information and Communications Technology (NICT)
3-5 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0289 Japan

Abstract

Machine transliteration is a method for automatically converting words in one language into phonetically equivalent ones in another language. Machine transliteration plays an important role in natural language applications such as information retrieval and machine translation, especially for handling proper nouns and technical terms. Four machine transliteration models – grapheme-based transliteration model, phoneme-based transliteration model, hybrid transliteration model, and correspondence-based transliteration model – have been proposed by several researchers. To date, however, there has been little research on a framework in which multiple transliteration models can operate simultaneously. Furthermore, there has been no comparison of the four models within the same framework and using the same data. We addressed these problems by 1) modeling the four models within the same framework, 2) comparing them under the same conditions, and 3) developing a way to improve machine transliteration through this comparison. Our comparison showed that the hybrid and correspondence-based models were the most effective and that the four models can be used in a complementary manner to improve machine transliteration performance.

1. Introduction

With the advent of new technology and the flood of information through the Web, it has become increasingly common to adopt foreign words into one's language. This usually entails adjusting the adopted word's original pronunciation to follow the phonological rules of the target language, along with modification of its orthographical form. This phonetic "translation" of foreign words is called *transliteration*. For example, the English word *data* is transliterated into Korean as 'de-i-teo'[1] and into Japanese as 'de-e-ta'. Transliteration is particularly used to translate proper names and technical terms from languages

1. In this paper, target language transliterations are represented in their Romanized form with single quotation marks and hyphens between syllables.

using Roman alphabets into ones using non-Roman alphabets such as from English to Korean, Japanese, or Chinese. Because transliteration is one of the main causes of the out-of-vocabulary (OOV) problem, transliteration by means of dictionary lookup is impractical (Fujii & Tetsuya, 2001; Lin & Chen, 2002). One way to solve the OOV problem is to use machine transliteration. Machine transliteration is usually used to support machine translation (MT) (Knight & Graehl, 1997; Al-Onaizan & Knight, 2002) and cross-language information retrieval (CLIR) (Fujii & Tetsuya, 2001; Lin & Chen, 2002). For CLIR, machine transliteration bridges the gap between the transliterated localized form and its original form by generating all possible transliterations from the original form (or generating all possible original forms from the transliteration)[2]. For example, machine transliteration can assist query translation in CLIR, where proper names and technical terms frequently appear in source language queries. In the area of MT, machine transliteration helps preventing translation errors when translations of proper names and technical terms are not registered in the translation dictionary. Machine transliteration can therefore improve the performance of MT and CLIR.

Four machine transliteration models have been proposed by several researchers: **grapheme[3]-based transliteration model** (ψ_G) (Lee & Choi, 1998; Jeong, Myaeng, Lee, & Choi, 1999; Kim, Lee, & Choi, 1999; Lee, 1999; Kang & Choi, 2000; Kang & Kim, 2000; Kang, 2001; Goto, Kato, Uratani, & Ehara, 2003; Li, Zhang, & Su, 2004), **phoneme[4]-based transliteration model** (ψ_P) (Knight & Graehl, 1997; Lee, 1999; Jung, Hong, & Paek, 2000; Meng, Lo, Chen, & Tang, 2001), **hybrid transliteration model** (ψ_H) (Lee, 1999; Al-Onaizan & Knight, 2002; Bilac & Tanaka, 2004), and **correspondence-based transliteration model** (ψ_C) (Oh & Choi, 2002). These models are classified in terms of the units to be transliterated. The ψ_G is sometimes referred to as the *direct method* because it directly transforms source language graphemes into target language graphemes without any phonetic knowledge of the source language words. The ψ_P is sometimes referred to as the *pivot method* because it uses source language phonemes as a pivot when it produces target language graphemes from source language graphemes. The ψ_P therefore usually needs two steps: 1) produce source language phonemes from source language graphemes; 2) produce target language graphemes from source phonemes[5]. The ψ_H and ψ_C make use of both source language graphemes and source language phonemes when producing target language transliterations. Hereafter, we refer to a source language grapheme as a *source*

2. The former process is generally called "transliteration", and the latter is generally called "back-transliteration" (Knight & Graehl, 1997)
3. *Graphemes* refer to the basic units (or the smallest contrastive units) of a written language: for example, English has 26 graphemes or letters, Korean has 24, and German has 30.
4. *Phonemes* are the simplest significant unit of sound (or the smallest contrastive units of a spoken language); for example, /M/, /AE/, and /TH/ in /M AE TH/, the pronunciation of *math*. We use the *ARPAbet* symbols to represent source phonemes. *ARPAbet* is one of the methods used for coding source phonemes into ASCII characters (`http://www.cs.cmu.edu/~laura/pages/arpabet.ps`). Here we denote source phonemes and pronunciation with two slashes, as in /AH/, and use pronunciation based on *The CMU Pronunciation Dictionary* and *The American Heritage(r) Dictionary of the English Language*.
5. These two steps are explicit if the transliteration system produces target language transliterations after producing the pronunciations of the source language words; they are implicit if the system uses phonemes implicitly in the transliteration stage and explicitly in the learning stage, as described elsewhere (Bilac & Tanaka, 2004)

grapheme, a source language phoneme as a *source phoneme*, and a target language grapheme as a *target grapheme*.

The transliterations produced by the four models usually differ because the models use different information. Generally, transliteration is a phonetic process, as in ψ_P, rather than an orthographic one, as in ψ_G (Knight & Graehl, 1997). However, standard transliterations are not restricted to phoneme-based transliterations. For example, the standard Korean transliterations of *data*, *amylase*, and *neomycin* are, respectively, the phoneme-based transliteration 'de-i-teo', the grapheme-based transliteration 'a-mil-la-a-je', and 'ne-o-ma-i-sin', which is a combination of the grapheme-based transliteration 'ne-o' and the phoneme-based transliteration 'ma-i-sin'. Furthermore, if the unit to be transliterated is restricted to either a source grapheme or a source phoneme, it is hard to produce the correct transliteration in many cases. For example, ψ_P cannot easily produce the grapheme-based transliteration 'a-mil-la-a-je', the standard Korean transliteration of *amylase*, because ψ_P tends to produce 'a-mil-le-i-seu' based on the sequence of source phonemes /AE M AH L EY S/. Multiple transliteration models should therefore be applied to better cover the various transliteration processes. To date, however, there has been little published research regarding a framework in which multiple transliteration models can operate simultaneously. Furthermore, there has been no reported comparison of the transliteration models within the same framework and using the same data although many English-to-Korean transliteration methods based on ψ_G have been compared to each other with the same data (Kang & Choi, 2000; Kang & Kim, 2000; Oh & Choi, 2002).

To address these problems, we 1) **modeled a framework in which the four transliteration models can operate simultaneously**, 2) **compared the transliteration models under the same conditions**, and 3) **using the results of the comparison, developed a way to improve the performance of machine transliteration**.

The rest of this paper is organized as follows. Section 2 describes previous work relevant to our study. Section 3 describes our implementation of the four transliteration models. Section 4 describes our testing and results. Section 5 describes a way to improve machine transliteration based on the results of our comparison. Section 6 describes a transliteration ranking method that can be used to improve transliteration performance. Section 7 concludes the paper with a summary and a look at future work.

2. Related Work

Machine transliteration has received significant research attention in recent years. In most cases, the source language and target language have been English and an Asian language, respectively – for example, English to Japanese (Goto et al., 2003), English to Chinese (Meng et al., 2001; Li et al., 2004), and English to Korean (Lee & Choi, 1998; Kim et al., 1999; Jeong et al., 1999; Lee, 1999; Jung et al., 2000; Kang & Choi, 2000; Kang & Kim, 2000; Kang, 2001; Oh & Choi, 2002). In this section, we review previous work related to the four transliteration models.

2.1 Grapheme-based Transliteration Model

Conceptually, the ψ_G is direct orthographical mapping from source graphemes to target graphemes. Several transliteration methods based on this model have been proposed, such as those based on a source-channel model (Lee & Choi, 1998; Lee, 1999; Jeong et al., 1999; Kim et al., 1999), a decision tree (Kang & Choi, 2000; Kang, 2001), a transliteration network (Kang & Kim, 2000; Goto et al., 2003), and a joint source-channel model (Li et al., 2004).

The methods based on the source-channel model deal with English-Korean transliteration. They use a chunk of graphemes that can correspond to a source phoneme. First, English words are segmented into a chunk of English graphemes. Next, all possible chunks of Korean graphemes corresponding to the chunk of English graphemes are produced. Finally, the most relevant sequence of Korean graphemes is identified by using the source-channel model. The advantage of this approach is that it considers a chunk of graphemes representing a phonetic property of the source language word. However, errors in the first step (segmenting the English words) propagate to the subsequent steps, making it difficult to produce correct transliterations in those steps. Moreover, there is high time complexity because all possible chunks of graphemes are generated in both languages.

In the method based on a decision tree, decision trees that transform each source grapheme into target graphemes are learned and then directly applied to machine transliteration. The advantage of this approach is that it considers a wide range of contextual information, say, the left three and right three contexts. However, it does not consider any phonetic aspects of transliteration.

Kang and Kim (2000) and Goto *et al.* (2003) proposed methods based on a transliteration network for, respectively, English-to-Korean and English-to-Japanese transliteration. Their frameworks for constructing a transliteration network are similar – both are composed of nodes and arcs. A node represents a chunk of source graphemes and its corresponding target graphemes. An arc represents a possible link between nodes and has a weight showing its strength. Like the methods based on the source-channel model, their methods consider the phonetic aspect in the form of chunks of graphemes. Furthermore, they segment a chunk of graphemes and identify the most relevant sequence of target graphemes in one step. This means that errors are not propagated from one step to the next, as in the methods based on the source-channel model.

The method based on the joint source-channel model simultaneously considers the source language and target language contexts (bigram and trigram) for machine transliteration. Its main advantage is the use of bilingual contexts.

2.2 Phoneme-based Transliteration Model

In the ψ_P, the transliteration key is pronunciation or the source phoneme rather than spelling or the source grapheme. This model is basically *source grapheme-to-source phoneme* transformation and *source phoneme-to-target grapheme* transformation.

Knight and Graehl (1997) modeled Japanese-to-English transliteration with weighted finite state transducers (WFSTs) by combining several parameters including romaji-to-phoneme, phoneme-to-English, English word probabilities, and so on. A similar model was developed for Arabic-to-English transliteration (Stalls & Knight, 1998). Meng *et al.* (2001) proposed an English-to-Chinese transliteration method based on English grapheme-to-phoneme conversion, cross-lingual phonological rules, mapping rules between English phonemes and Chinese phonemes, and Chinese syllable-based and character-based language models. Jung *et al.* (2000) modeled English-to-Korean transliteration with an extended Markov window. The method transforms an English word into English pronunciation by using a pronunciation dictionary. Then it segments the English phonemes into chunks of English phonemes; each chunk corresponds to a Korean grapheme as defined by handcrafted rules. Finally, it automatically transforms each chunk of English phonemes into Korean graphemes by using an extended Markov window.

Lee (1999) modeled English-to-Korean transliteration in two steps. The *English grapheme-to-English phoneme* transformation is modeled in a manner similar to his method based on the source-channel model described in Section 2.1. The English phonemes are then transformed into Korean graphemes by using English-to-Korean standard conversion rules (EKSCR) (Korea Ministry of Culture & Tourism, 1995). These rules are in the form of context-sensitive rewrite rules, "$P_A P_X P_B \rightarrow y$", meaning that English phoneme P_X is rewritten as Korean grapheme y in the context P_A and P_B, where P_X, P_A, and P_B represent English phonemes. For example, "$P_A = *, P_X = /SH/, P_B = end \rightarrow$ 'si'" means "English phoneme $/SH/$ is rewritten into Korean grapheme 'si' if it occurs at the end of the word (end) after any phoneme ($*$)". This approach suffers from both the propagation of errors and the limitations of EKSCR. The first step, grapheme-to-phoneme transformation, usually results in errors, and the errors propagate to the next step. Propagated errors make it difficult for a transliteration system to work correctly. In addition, EKSCR does not contain enough rules to generate correct Korean transliterations since its main focus is mapping from an English phoneme to Korean graphemes without taking into account the contexts of the English grapheme.

2.3 Hybrid and Correspondence-based Transliteration Models

Attempts to use both source graphemes and source phonemes in machine transliteration led to the correspondence-based transliteration model (ψ_C) (Oh & Choi, 2002) and the hybrid transliteration model (ψ_H) (Lee, 1999; Al-Onaizan & Knight, 2002; Bilac & Tanaka, 2004). The former makes use of the correspondence between a source grapheme and a source phoneme when it produces target language graphemes; the latter simply combines ψ_G and ψ_P through linear interpolation. Note that the ψ_H combines the grapheme-based transliteration probability ($Pr(\psi_G)$) and the phoneme-based transliteration probability ($Pr(\psi_P)$) using linear interpolation.

Oh and Choi (2002) considered the contexts of a source grapheme and its corresponding source phoneme for English-to-Korean transliteration. They used EKSCR as the basic rules in their method. Additional contextual rules are semi-automatically constructed by examining the cases in which EKSCR produced incorrect transliterations because of a lack of contexts. These contextual rules are in the form of context-sensitive rewrite rules, "$C_A C_X C_B \rightarrow y$", meaning "$C_X$ is rewritten as target grapheme y in the context C_A and C_B". Note that C_X, C_A, and C_B represent the correspondence between the English grapheme and phoneme. For example, we can read "$C_A = (* : /Vowel/), C_X = (r : /R/), C_B = (* : /Consonant/) \rightarrow$ NULL" as "English grapheme r corresponding to phoneme $/R/$ is rewritten into null Korean graphemes when it occurs after vowel phonemes, $(* : /Vowel/)$, before consonant phonemes, $(* : /Consonant/)$". The main advantage of this approach is the application of a sophisticated rule that reflects the context of the source grapheme and source phoneme by considering their correspondence. However, there is lack of portability to other languages because the rules are restricted to Korean.

Several researchers (Lee, 1999; Al-Onaizan & Knight, 2002; Bilac & Tanaka, 2004) have proposed hybrid model-based transliteration methods. They model ψ_G and ψ_P with WFSTs or a source-channel model and combine ψ_G and ψ_P through linear interpolation. In their ψ_P, several parameters are considered, such as the *source grapheme-to-source phoneme* probability, *source phoneme-to-target grapheme* probability, and *target language word* probability. In their ψ_G, the *source grapheme-to-target grapheme* probability is mainly considered. The main disadvantage of the hybrid model is that the dependence between the source grapheme and source phoneme is not taken into consideration in the combining process; in contrast, Oh and Choi's approach (Oh & Choi, 2002) considers this dependence by using the correspondence between the source grapheme and phoneme.

3. Modeling Machine Transliteration Models

In this section, we describe our implementation of the four machine transliteration models (ψ_G, ψ_P, ψ_H, and ψ_C) using three machine learning algorithms: memory-based learning, decision-tree learning, and the maximum entropy model.

3.1 Framework for Four Machine Transliteration Models

Figure 1 summarizes the differences among the transliteration models and their component functions. The ψ_G directly transforms source graphemes (S) into target graphemes (T). The ψ_P and ψ_C transform source graphemes into source phonemes and then generate target graphemes[6]. While ψ_P uses only the source phonemes, ψ_C uses the correspondence between the source grapheme and the source phoneme when it generates target graphemes. We describe their differences with two functions, ϕ_{PT} and $\phi_{(SP)T}$. The ψ_H is represented as the linear interpolation of $Pr(\psi_G)$ and $Pr(\psi_P)$ by means of α ($0 \leq \alpha \leq 1$). Here, $Pr(\psi_P)$ is the probability that ψ_P will produce target graphemes, while $Pr(\psi_G)$ is the probability that ψ_G will produce target graphemes. We can thus regard ψ_H as being composed of component functions of ψ_G and ψ_P (ϕ_{SP}, ϕ_{PT}, and ϕ_{ST}). Here we use the maximum entropy model as the machine learning algorithm for ψ_H because ψ_H requires $Pr(\psi_P)$ and $Pr(\psi_G)$, and only the maximum entropy model among memory-based learning, decision-tree learning, and the maximum entropy model can produce the probabilities.

To train each component function, we need to define the features that represent training instances and data. Table 1 shows five feature types, f_S, f_P, f_{Stype}, f_{Ptype}, and f_T. The feature types used depend on the component functions. The modeling of each component function with the feature types is explained in Sections 3.2 and 3.3.

3.2 Component Functions of Each Transliteration Model

Table 2 shows the definitions of the four component functions that we used. Each is defined in terms of its input and output: the first and last characters in the notation of each correspond respectively to its input and output. The role of each component function in

6. According to $(g \circ f)(x) = g(f(x))$, we can write $(\phi_{(SP)T} \circ \phi_{SP})(x) = \phi_{(SP)T}(\phi_{SP}(x))$ and $(\phi_{PT} \circ \phi_{SP})(x) = \phi_{PT}(\phi_{SP}(x))$.

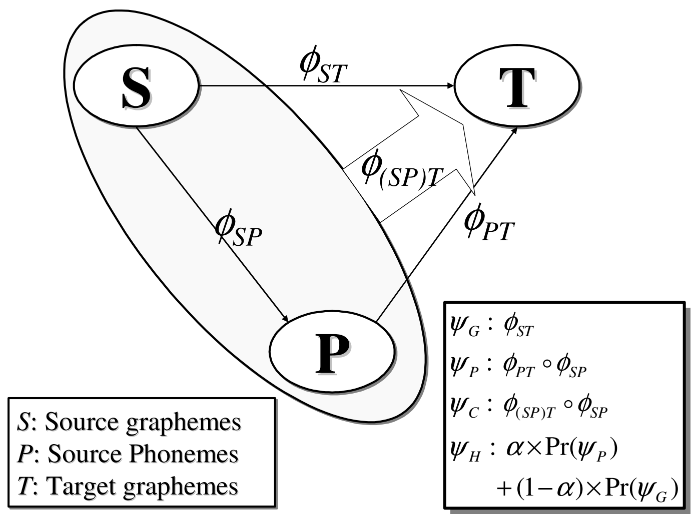


Figure 1: Graphical representation of each component function and four transliteration models: S is a set of source graphemes (e.g., letters of the English alphabet), P is a set of source phonemes defined in *ARPAbet*, and T is a set of target graphemes.

Feature type		Description and possible values
$f_{S,Stype}$	f_S	Source graphemes in S: 26 letters in English alphabet
	f_{Stype}	Source grapheme types: Consonant (C) and Vowel (V)
$f_{P,Ptype}$	f_P	Source phonemes in P (/AA/, /AE/, and so on)
	f_{Ptype}	Source phoneme types: Consonant (C), Vowel (V), Semi-vowel (SV), and silence (/∼/)
f_T		Target graphemes in T

Table 1: Feature types used for transliteration models: $f_{S,Stype}$ indicates both f_S and f_{Stype}, while $f_{P,Ptype}$ indicates both f_P and f_{Ptype}.

each transliteration model is to produce the most relevant output from its input. The performance of a transliteration model therefore depends strongly on that of its component functions. In other words, the better the modeling of each component function, the better the performance of the machine transliteration system.

The modeling strongly depends on the feature type. Different feature types are used by the $\phi_{(SP)T}$, ϕ_{PT}, and ϕ_{ST} functions, as shown in Table 2. These three component functions thus have different strengths and weaknesses for machine transliteration. The ϕ_{ST} function is good at producing grapheme-based transliterations and poor at producing

Notation	Feature types used	Input	Output
ϕ_{SP}	$f_{S,Stype}$, f_P	$s_i, c(s_i)$	p_i
$\phi_{(SP)T}$	$f_{S,Stype}$, $f_{P,Ptype}$, f_T	$s_i, p_i, c(s_i), c(p_i)$	t_i
ϕ_{PT}	$f_{P,Ptype}$, f_T	$p_i, c(p_i)$	t_i
ϕ_{ST}	$f_{S,Stype}$, f_T	$s_i, c(s_i)$	t_i

Table 2: Definition of each component function: s_i, $c(s_i)$, p_i, $c(p_i)$, and t_i respectively represent the i^{th} source grapheme, the context of s_i ($s_{i-n}, \cdots, s_{i-1}$ and $s_{i+1}, \cdots, s_{i+n}$), the i^{th} source phoneme, the context of p_i ($p_{i-n}, \cdots, p_{i-1}$ and $p_{i+1}, \cdots, p_{i+n}$), and the i^{th} target grapheme.

phoneme-based ones. In contrast, the ϕ_{PT} function is good at producing phoneme-based transliterations and poor at producing grapheme-based ones. For *amylase* and its standard Korean transliteration, 'a-mil-la-a-je', which is a grapheme-based transliteration, ϕ_{ST} tends to produce the correct transliteration; ϕ_{PT} tends to produce wrong ones like 'ae-meol-le-i-seu', which is derived from /AE M AH L EY S/, the pronunciation of *amylase.* In contrast, ϕ_{PT} can produce 'de-i-teo', which is the standard Korean transliteration of *data* and a phoneme-based transliteration, while ϕ_{ST} tends to give a wrong one, like 'da-ta'.

The $\phi_{(SP)T}$ function combines the advantages of ϕ_{ST} and ϕ_{PT} by utilizing the correspondence between the source grapheme and source phoneme. This correspondence enables $\phi_{(SP)T}$ to produce both grapheme-based and phoneme-based transliterations. Furthermore, the correspondence provides important clues for use in resolving transliteration ambiguities[7]. For example, the source phoneme /AH/ produces much ambiguity in machine transliteration because it can be mapped to almost every vowel in the source and target languages (the underlined graphemes in the following example corresponds to /AH/: hol<u>o</u>caust in English, 'hol-l<u>o</u>-ko-seu-teu' in its Korean counterpart, and 'ho-r<u>o</u>-ko-o-su-to' in its Japanese counterpart). If we know the correspondence between the source grapheme and source phoneme, we can more easily infer the correct transliteration of /AH/ because the correct target grapheme corresponding to /AH/ usually depends on the source grapheme corresponding to /AH/. Moreover, there are various Korean transliterations of the source grapheme *a*: 'a', 'ae', 'ei', 'i', and 'o'. In this case, the English phonemes corresponding to the English grapheme can help a component function resolve transliteration ambiguities, as shown in Table 3. In Table 3, the *a* underlined in the example words shown in the last column is pronounced as the English phoneme in the second column. By looking at English grapheme and its corresponding English phoneme, we can find correct Korean transliterations more easily.

Though $\phi_{(SP)T}$ is more effective than both ϕ_{ST} and ϕ_{PT} in many cases, $\phi_{(SP)T}$ sometimes works poorly when the standard transliteration is strongly biased to either grapheme-based or phoneme-based transliteration. In such cases, either the source grapheme or source phoneme does not contribute to the correct transliteration, making it difficult for $\phi_{(SP)T}$ to produce the correct transliteration. Because ϕ_{ST}, ϕ_{PT}, and $\phi_{(SP)T}$ are the core parts

7. Though contextual information can also be used to reduce ambiguities, we limit our discussion here to the feature type.

Korean Grapheme	English Phoneme	Example usage
‘a’	/AA/	adagio, safari, vivace
‘ae’	/AE/	advantage, alabaster, travertine
‘ei’	/EY/	chamber, champagne, chaos
‘i’	/IH/	advantage, average, silage
‘o’	/AO/	allspice, ball, chalk

Table 3: Examples of Korean graphemes derived from English grapheme *a* and its corresponding English phonemes: the underlines in the example words indicate the English grapheme corresponding to English phonemes in the second column.

of ψ_G, ψ_P, and ψ_C, respectively, the advantages and disadvantages of the three component functions correspond to those of the transliteration models in which each is used.

Transliteration usually depends on context. For example, the English grapheme *a* can be transliterated into Korean graphemes on the basis of its context, like ‘ei’ in the context of *-ation* and ‘a’ in the context of *art*. When context information is used, determining the context window size is important. A context window that is too narrow can degrade transliteration performance because of a lack of context information. For example, when English grapheme *t* in *-tion* is transliterated into Korean, the one right English grapheme is insufficient as context because the three right contexts, *-ion*, are necessary to get the correct Korean grapheme, ‘s’. A context window that is too wide can also degrade transliteration performance because it reduces the power to resolve transliteration ambiguities. Many previous studies have determined that an appropriate context window size is 3. In this paper, we use a window size of 3, as in previous work (Kang & Choi, 2000; Goto et al., 2003). The effect of the context window size on transliteration performance will be discussed in Section 4.

Table 4 shows how to identify the most relevant output in each component function using context information. The L3-L1, C0, and R1-R3 represent the left context, current context (i.e., that to be transliterated), and right context, respectively. The ϕ_{SP} function produces the most relevant source phoneme for each source grapheme. If $SW = s_1 \cdot s_2 \cdot \ldots \cdot s_n$ is an English word, SW's pronunciation can be represented as a sequence of source phonemes produced by ϕ_{SP}; that is, $P_{SW} = p_1 \cdot p_2 \cdot \ldots \cdot p_n$, where $p_i = \phi_{SP}(s_i, c(s_i))$. ϕ_{SP} transforms source graphemes into phonemes in two ways. The first one is to search in a pronunciation dictionary containing English words and their pronunciation (CMU, 1997). The second one is to estimate the pronunciation (or automatic grapheme-to-phoneme conversion) (Andersen, Kuhn, Lazarides, Dalsgaard, Haas, & Noth, 1996; Daelemans & van den Bosch, 1996; Pagel, Lenzo, & Black, 1998; Damper, Marchand, Adamson, & Gustafson, 1999; Chen, 2003). If an English word is not registered in the pronunciation dictionary, we must estimate its pronunciation. The produced pronunciation is used for ϕ_{PT} in ψ_P and $\phi_{(SP)T}$ in ψ_C. For training the automatic grapheme-to-phoneme conversion in ϕ_{SP}, we use *The CMU Pronouncing Dictionary* (CMU, 1997).

The ϕ_{ST}, ϕ_{PT}, and $\phi_{(SP)T}$ functions produce target graphemes using their input. Like ϕ_{SP}, these three functions use their previous outputs, which are represented by f_T. As

	Type	L3	L2	L1	C0	R1	R2	R3		Output
ϕ_{SP}	f_S	\$	\$	\$	b	o	a	r	$\rightarrow$	/B/
	f_{Stype}	\$	\$	\$	C	V	V	C		
	f_P	\$	\$	\$	ϵ					
ϕ_{ST}	f_S	\$	\$	\$	b	o	a	r	$\rightarrow$	'b'
	f_{Stype}	\$	\$	\$	C	V	V	C		
	f_T	\$	\$	\$	ϵ					
ϕ_{PT}	f_P	\$	\$	\$	/B/	/AO/	/∼/	/R/	$\rightarrow$	'b'
	f_{Ptype}	\$	\$	\$	C	V	/∼/	C		
	f_T	\$	\$	\$	ϵ					
$\phi_{(SP)T}$	f_S	\$	\$	\$	b	o	a	r	$\rightarrow$	'b'
	f_P	\$	\$	\$	/B/	/AO/	/∼/	/R/		
	f_{Stype}	\$	\$	\$	C	V	V	C		
	f_{Ptype}	\$	\$	\$	C	V	/∼/	C		
	f_T	\$	\$	\$	ϵ					

Table 4: Framework for each component function: \$ represents start of words and ϵ means unused contexts for each component function.

shown in Table 4, ϕ_{ST}, ϕ_{PT}, and $\phi_{(SP)T}$ produce target grapheme 'b' for source grapheme *b* and source phoneme /B/ in *b̲oard* and /B̲ AO R D/. Because the *b* and /B/ are the first source grapheme of *b̲oard* and the first source phoneme of /B̲ AO R D/, respectively, their left context is \$, which represents the start of words. Source graphemes (*o*, *a*, and *r*) and their type (V: vowel, V: vowel, and C: consonant) can be the right context in ϕ_{ST} and $\phi_{(SP)T}$. Source phonemes (/AO/, /∼/, and /R/) and their type (V: vowel, /∼/: silence, V: vowel) can be the right context in ϕ_{PT} and $\phi_{(SP)T}$. Depending on the feature type used in each component function and described in Table 2, ϕ_{ST}, ϕ_{PT}, and $\phi_{(SP)T}$ produce a sequence of target graphemes, $T_{SW} = t_1 \cdot t_2 \cdot \ldots \cdot t_n$, for $SW = s_1 \cdot s_2 \cdot \ldots \cdot s_n$ and $P_{SW} = p_1 \cdot p_2 \cdot \ldots \cdot p_n$. For *board*, SW, P_{SW}, and T_{SW} can be represented as follows. The /∼/ represents silence (null source phonemes), and the '∼' represents null target graphemes.

- $SW = s_1 \cdot s_2 \cdot s_3 \cdot s_4 \cdot s_5 = b \cdot o \cdot a \cdot r \cdot d$
- $P_{SW} = p_1 \cdot p_2 \cdot p_3 \cdot p_4 \cdot p_5 = /B/ \cdot /AO/ \cdot / \sim / \cdot /R/ \cdot /D/$
- $T_{SW} = t_1 \cdot t_2 \cdot t_3 \cdot t_4 \cdot t_5 =$ 'b'$\cdot$ 'o' $\cdot$ '∼' $\cdot$ '∼' $\cdot$ 'deu'

3.3 Machine Learning Algorithms for Each Component Function

In this section we describe a way to model component functions using three machine learning algorithms (the maximum entropy model, decision-tree learning, and memory-based learning)[8]. Because the four component functions share a similar framework, we limit our focus to $\phi_{(SP)T}$ in this section.

8. These three algorithms are typically applied to automatic grapheme-to-phoneme conversion (Andersen et al., 1996; Daelemans & van den Bosch, 1996; Pagel et al., 1998; Damper et al., 1999; Chen, 2003).

3.3.1 Maximum entropy model

The maximum entropy model (MEM) is a widely used probability model that can incorporate heterogeneous information effectively (Berger, Pietra, & Pietra, 1996). In the MEM, an event (ev) is usually composed of a target event (te) and a history event (he); say $ev =< te, he >$. Event ev is represented by a bundle of feature functions, $fe_i(ev)$, which represent the existence of certain characteristics in event ev. A feature function is a binary-valued function. It is activated ($fe_i(ev) = 1$) when it meets its activating condition; otherwise it is deactivated ($fe_i(ev) = 0$) (Berger et al., 1996). Let source language word SW be composed of n graphemes. SW, P_{SW}, and T_{SW} can then be represented as $SW = s_1, \cdots, s_n$, $P_{SW} = p_1, \cdots, p_n$, and $T_{SW} = t_1, \cdots, t_n$, respectively. P_{SW} and T_{SW} represent the pronunciation and target language word corresponding to SW, and p_i and t_i represent the source phoneme and target grapheme corresponding to s_i. Function $\phi_{(SP)T}$ based on the maximum entropy model can be represented as

$$Pr(T_{SW}|SW, P_{SW}) = Pr(t_1, \cdots, t_n|s_1, \cdots, s_n, p_1, \cdots, p_n) \tag{1}$$

With the assumption that $\phi_{(SP)T}$ depends on the context information in window size k, we simplify Formula (1) to

$$Pr(T_{SW}|SW, P_{SW}) \approx \prod_i Pr(t_i|t_{i-k}, \cdots, t_{i-1}, p_{i-k}, \cdots, p_{i+k}, s_{i-k}, \cdots, s_{i+k}) \tag{2}$$

Because $t_1, \cdots, t_n$, $s_1, \cdots, s_n$, and $p_1, \cdots, p_n$ can be represented by f_T, $f_{S,Stype}$, and $f_{P,Ptype}$, respectively, we can rewrite Formula (2) as

$$Pr(T_{SW}|SW, P_{SW}) \approx \prod_i Pr(t_i|f_{T_{(i-k,i-1)}}, f_{P,Ptype_{(i-k,i+k)}}, f_{S,Stype_{(i-k,i+k)}}) \tag{3}$$

where i is the index of the current source grapheme and source phoneme to be transliterated and $f_{X(l,m)}$ represents the features of feature type f_X located from position l to position m.

An important factor in designing a model based on the maximum entropy model is to identify feature functions that effectively support certain decisions of the model. Our basic philosophy of feature function design for each component function is that the context information collocated with the unit of interest is important. We thus designed the feature function with collocated features in each feature type and between different feature types. Features used for $\phi_{(SP)T}$ are listed below. These features are used as activating conditions or history events of feature functions.

- Feature type and features used for designing feature functions in $\phi_{(SP)T}$ ($k = 3$)
  - All possible features in $f_{S,Stype_{i-k,i+k}}$, $f_{P,Ptype_{i-k,i+k}}$, and $f_{T_{i-k,i-1}}$ (e.g., $f_{S_{i-1}}$, $f_{P_{i-1}}$, and $f_{T_{i-1}}$)
  - All possible feature combinations between features of the same feature type (e.g., $\{f_{S_{i-2}}, f_{S_{i-1}}, f_{S_{i+1}}\}$, $\{f_{P_{i-2}}, f_{P_i}, f_{P_{i+2}}\}$, and $\{f_{T_{i-2}}, f_{T_{i-1}}\}$)
  - All possible feature combinations between features of different feature types (e.g., $\{f_{S_{i-1}}, f_{P_{i-1}}\}$, $\{f_{S_{i-1}}, f_{T_{i-2}}\}$, and $\{f_{Ptype_{i-2}}, f_{P_{i-3}}, f_{T_{i-2}}\}$)
    - between $f_{S,Stype_{i-k,i+k}}$ and $f_{P,Ptype_{i-k,i+k}}$

fe_j	te	he		
	t_i	$f_{T_{(i-k,i-1)}}$	$f_{S,Stype_{(i-k,i+k)}}$	$f_{P,Ptype_{(i-k,i+k)}}$
fe_1	‘b’	–	$f_{S_i} = b$	$f_{P_i} = /B/$
fe_2	‘b’	–	$f_{S_{i-1}} = \$$	–
fe_3	‘b’	$f_{T_{i-1}} = \$$	$f_{S_{i+1}} = o$ and $f_{Stype_{i+2}} = V$	$f_{P_i} = /B/$
fe_4	‘b’	–	–	$f_{P_{i+1}} = /AO/$
fe_5	‘b’	$f_{T_{i-2}} = \$$	$f_{S_{i+3}} = r$	$f_{Ptype_i} = C$

Table 5: Feature functions for $\phi_{(SP)T}$ derived from Table 4.

* between $f_{S,Stype_{i-k,i+k}}$ and $f_{T_{i-k,i-1}}$
* between $f_{P,Ptype_{i-k,i+k}}$ and $f_{T_{i-k,i-1}}$

Generally, a conditional maximum entropy model that gives the conditional probability $Pr(y|x)$ is represented as Formula (4) (Berger et al., 1996).

$$\begin{aligned} Pr(y|x) &= \frac{1}{Z(x)} exp(\sum_i \lambda_i fe_i(x,y)) \\ Z(x) &= \sum_y exp(\sum_i \lambda_i fe_i(x,y)) \end{aligned} \tag{4}$$

In $\phi_{(SP)T}$, the target event (te) is target graphemes to be assigned, and the history event (he) can be represented as a tuple $< f_{T_{(i-k,i-1)}}, f_{S,Stype_{(i-k,i+k)}}, f_{P,Ptype_{(i-k,i+k)}} >$. Therefore, we can rewrite Formula (3) as

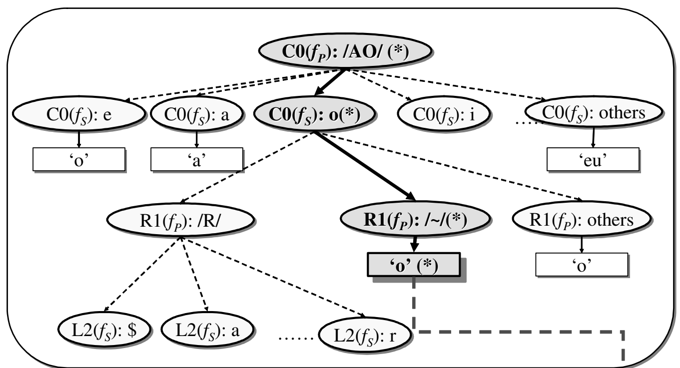
$$\begin{aligned} & Pr(t_i|f_{T_{(i-k,i-1)}}, f_{S,Stype_{(i-k,i+k)}}, f_{P,Ptype_{(i-k,i+k)}}) \\ & = Pr(te|he) = \frac{1}{Z(he)} exp(\sum_i \lambda_i fe_i(he,te)) \end{aligned} \tag{5}$$

Table 5 shows example feature functions for $\phi_{(SP)T}$; Table 4 was used to derive the functions. For example, fe_1 represents an event where he (history event) is “f_{S_i} is b and f_{P_i} is /B/” and te (target event) is “f_{T_i} is ‘b’”. To model each component function based on the MEM, Zhang’s maximum entropy modeling tool is used (Zhang, 2004).

3.3.2 Decision-tree learning

Decision-tree learning (DTL) is one of the most widely used and well-known methods for inductive inference (Quinlan, 1986; Mitchell, 1997). ID3, which is a greedy algorithm that constructs decision trees in a top-down manner, uses the information gain, which is a measure of how well a given feature (or attribute) separates training examples on the basis of their target class (Quinlan, 1993; Manning & Schutze, 1999). We use C4.5 (Quinlan, 1993), which is a well-known tool for DTL and an implementation of Quinlan’s ID3 algorithm.

The training data for each component function is represented by features located in L3-L1, C0, and R1-R3, as shown in Table 4. C4.5 tries to construct a decision tree by looking for regularities in the training data (Mitchell, 1997). Figure 2 shows part of the decision tree constructed for $\phi_{(SP)T}$ in English-to-Korean transliteration. A set of the target classes in the decision tree for $\phi_{(SP)T}$ is a set of the target graphemes. The rectangles indicate the leaf nodes, which represent the target classes, and the circles indicate the decision nodes. To simplify our examples, we use only f_S and f_P. Note that all feature types for each component function, as described in Table 4, are actually used to construct decision trees. Intuitively, the most effective feature from among L3-L1, C0, and R1-R3 for $\phi_{(SP)T}$ may be located in C0 because the correct outputs of $\phi_{(SP)T}$ strongly depend on the source grapheme or source phoneme in the C0 position. As we expected, the most effective feature in the decision tree is located in the C0 position, that is, C0(f_P). (Note that the first feature to be tested in decision trees is the most effective feature.) In Figure 2, the decision tree produces the target grapheme (Korean grapheme) 'o' for the instance $x(SPT)$ by retrieving the decision nodes from $C0(f_P) = /AO/$ to $R1(f_P) = / \sim /$ represented by '$*$'.



x(SPT)	Feature type	L3	L2	L1	C0	R1	R2	R3		$\phi_{(SP)T}$
	f_S	\$	\$	b	o	a	r	d	→	**'o'**
	f_P	\$	\$	/B/	/AO/	/~/	/R/	/D/		

Figure 2: Decision tree for $\phi_{(SP)T}$.

3.3.3 Memory-based learning

Memory-based learning (MBL), also called "instance-based learning" and "case-based learning", is an example-based learning method. It is based on a k-nearest neighborhood algorithm (Aha, Kibler, & Albert, 1991; Aha, 1997; Cover & Hart, 1967; Devijver & Kittler., 1982). MBL represents training data as a vector and, in the training phase, it places all training data as examples in memory and clusters some examples on the basis of the k-nearest neighborhood principle. Training data for MBL is represented in the same form as training data for a decision tree. Note that the target classes for $\phi_{(SP)T}$, which MBL outputs, are target graphemes. Feature weighting to deal with features of differing importance is also done in the training phase[9]. It then produces an output using similarity-based

9. TiMBL (Daelemans, Zavrel, Sloot, & Bosch, 2004) supports *gain ratio weighting, information gain weighting, chi-squared (χ^2) weighting*, and *shared variance weighting* of the features.

reasoning between test data and the examples in memory. If the test data is x and the set of examples in memory is Y, the similarity between x and Y can be estimated using distance function $\Delta(x, Y)$[10]. MBL selects an example y_i or the cluster of examples that are most similar to x and then assigns the example's target class to x's target class. We use an MBL tool called TiMBL (Tilburg memory-based learner) version 5.0 (Daelemans et al., 2004).

4. Experiments

We tested the four machine transliteration models on English-to-Korean and English-to-Japanese transliteration. The test set for the former (EKSet) (Nam, 1997) consisted of 7,172 English-Korean pairs – the number of training items was about 6,000 and that of the blind test items was about 1,000. EKSet contained no transliteration variations, meaning that there was one transliteration for each English word. The test set for the latter (EJSet) contained English-katakana pairs from EDICT (Breen, 2003) and consisted of 10,417 pairs – the number of training items was about 9,000 and that of the blind test items was about 1,000. EJSet contained transliteration variations, like <*micro*, 'ma-i-ku-ro'>, and <*micro*, 'mi-ku-ro'>; the average number of Japanese transliterations for an English word was 1.15. EKSet and EJSet covered proper names, technical terms, and general terms. We used *The CMU Pronouncing Dictionary* (CMU, 1997) for training pronunciation estimation (or automatic grapheme-to-phoneme conversion) in ϕ_{SP}. The training for automatic grapheme-to-phoneme conversion was done ignoring the lexical stress of vowels in the dictionary (CMU, 1997). The evaluation was done in terms of word accuracy (WA), the evaluation measure used in previous work (Kang & Choi, 2000; Kang & Kim, 2000; Goto et al., 2003; Bilac & Tanaka, 2004). Here, WA can be represented as Formula (6). A generated transliteration for an English word was judged to be correct if it exactly matched a transliteration for that word in the test data.

$$WA = \frac{number\ of\ correct\ transliterations\ output\ by\ system}{number\ of\ transliterations\ in\ blind\ test\ data} \tag{6}$$

In the evaluation, we used k-fold cross-validation (k=7 for EKSet and k=10 for EJSet). The test set was divided into k subsets. Each was used in turn for testing while the remainder was used for training. The average WA computed across all k trials was used as the evaluation results presented in this section.

We conducted six tests.

- Hybrid Model Test: Evaluation of hybrid transliteration model by changing value of α (the parameter of the hybrid transliteration model)
- Comparison Test I: Comparison among four machine transliteration models
- Comparison Test II: Comparison of four machine transliteration models to previously proposed transliteration methods

10. *Modified value difference metric, overlap metric, Jeffrey divergence metric, dot product metric*, etc. are used as the distance function (Daelemans et al., 2004).

- Dictionary Test: Evaluation of transliteration models on words registered and not registered in pronunciation dictionary to determine effect of pronunciation dictionary on each model

- Context Window-Size Test: Evaluation of transliteration models for various sizes of context window

- Training Data-Size Test: Evaluation of transliteration models for various sizes of training data sets

4.1 Hybrid Model Test

The objective of this test was to estimate the dependence of the performance of ψ_H on parameter α. We evaluated the performance by changing α from 0 to 1 at intervals of 0.1 (i.e., α=0, 0.1, 0.2, $\cdots$, 0.9, 1.0). Note that the hybrid model can be represented as "$\alpha \times Pr(\psi_P) + (1-\alpha) \times Pr(\psi_G)$". Therefore, ψ_H is ψ_G when $\alpha = 0$ and ψ_P when $\alpha = 1$. As shown in Table 6, the performance of ψ_H depended on that of ψ_G and ψ_P. For example, the performance of ψ_G exceeded that of ψ_P for EKSet. Therefore, ψ_H tended to perform better when $\alpha \leq 0.5$ than when $\alpha > 0.5$ for EKSet. The best performance was attained when $\alpha = 0.4$ for EKSet and when $\alpha = 0.5$ for EJSet. Hereinafter, we use $\alpha = 0.4$ for EKSet and $\alpha = 0.5$ for EJSet as the linear interpolation parameter for ψ_H.

α	EKSet	EJSet
0	58.8%	58.8%
0.1	61.2%	60.9%
0.2	62.0%	62.6%
0.3	63.0%	64.1%
0.4	64.1%	65.4%
0.5	63.4%	65.8%
0.6	61.1%	65.0%
0.7	59.6%	63.4%
0.8	58.2%	62.1%
0.9	57.0%	61.2%
1.0	55.2%	59.2%

Table 6: Results of Hybrid Model Test.

4.2 Comparison Test I

The objectives of the first comparison test were to compare performance among the four transliteration models (ψ_G, ψ_P, ψ_H, and ψ_C) and to compare the performance of each model with the combined performance of three of the models (ψ_{G+P+C}). Table 7 summarizes the performance of each model for English-to-Korean and English-to-Japanese transliteration, where DTL, MBL[11] and MEM represent decision-tree learning, memory-based learning, and maximum entropy model.

The unit to be transliterated was restricted to either a source grapheme or a source phoneme in ψ_G and ψ_P; it was dynamically selected on the basis of the contexts in ψ_H and ψ_C. This means that ψ_G and ψ_P could produce an incorrect result if either a source phoneme or a source grapheme, which, respectively, they do not consider, holds the key to producing the correct transliteration result. For this reason, ψ_H and ψ_C performed better than both ψ_G and ψ_P.

Transliteration Model	EKSet			EJSet		
	DTL	MBL	MEM	DTL	MBL	MEM
ψ_G	53.1%	54.6%	58.8%	55.6%	58.9%	58.8%
ψ_P	50.8%	50.6%	55.2%	55.8%	56.1%	59.2%
ψ_H	N/A	N/A	64.1%	N/A	N/A	65.8%
ψ_C	59.5%	60.3%	65.5%	64.0%	65.8%	69.1%
ψ_{G+P+C}	72.0%	71.4%	75.2%	73.4%	74.2%	76.6%

Table 7: Results of Comparison Test I.

In the table, ψ_{G+P+C} means the combined results for the three transliteration models, ψ_G, ψ_P, and ψ_C. We exclude ψ_H from the combining because it is implemented only with the MEM (the performance of combining the four transliteration models are discussed in Section 5). In evaluating ψ_{G+P+C}, we judged the transliteration results to be correct if there was at least one correct transliteration among the results produced by the three models. Though ψ_C showed the best results among the three transliteration models due to its ability to use the correspondence between the source grapheme and source phoneme, the source grapheme or the source phoneme can create noise when the correct transliteration is produced by the other one. In other words, when the correct transliteration is strongly biased to either grapheme-based or phoneme-based transliteration, ψ_G and ψ_P may be more suitable for producing the correct transliteration.

Table 8 shows example transliterations produced by each transliteration model. The ψ_G produced correct transliterations for *cyclase* and *bacteroid*, while ψ_P did the same for *geoid* and *silo*. ψ_C produced correct transliterations for *saxhorn* and *bacteroid*, and ψ_H produced correct transliterations for *geoid* and *bacteroid*. As shown by these results, there are transliterations that only one transliteration model can produce correctly. For example, only ψ_G, ψ_P, and ψ_C produced the correct transliterations of *cyclase*, *silo*, and *saxhorn*, respectively. Therefore, these three transliteration models can be used in a complementary manner to improve transliteration performance because at least one can usually produce the correct transliteration. This combination increased the performance by compared to ψ_G, ψ_P, and ψ_C (on average, 30.1% in EKSet and 24.6% in EJSet). In short, ψ_G, ψ_P, and ψ_C are complementary transliteration models that together produce more correct transliterations,

11. We tested all possible combinations between $\Delta(x, Y)$ and a weighting scheme supported by TiMBL (Daelemans et al., 2004) and did not detect any significant differences in performance for the various combinations. Therefore, we used the default setting of TiMBL (*Overlap metric* for $\Delta(x, Y)$ and *gain ratio weighting* for feature weighting).

so combining different transliteration models can improve transliteration performance. The transliteration results produced by ψ_{G+P+C} are analyzed in detail in Section 5.

	ψ_G	ψ_P
cyclase	'si-keul-la-a-je'	$*$'sa-i-keul-la-a-je'
bacteroid	'bak-te-lo-i-deu'	$*$'bak-teo-o-i-deu'
geoid	$*$'je-o-i-deu'	'ji-o-i-deu'
silo	$*$'sil-lo'	'sa-il-lo'
saxhorn	$*$'saek-seon'	$*$'saek-seu-ho-leun'
	ψ_H	ψ_C
cyclase	$*$'sa-i-keul-la-a-je'	$*$'sa-i-keul-la-a-je'
bacteroid	'bak-te-lo-i-deu'	'bak-te-lo-i-deu'
geoid	'ji-o-i-deu'	$*$'ge-o-i-deu'
silo	$*$'sil-lo'	$*$'sil-lo'
saxhorn	$*$'saek-seon'	'saek-seu-hon'

Table 8: Example transliterations produced by each transliteration model ($*$ indicates an incorrect transliteration).

In our subsequent testing, we used the maximum entropy model as the machine learning algorithm for two reasons. First, it produced the best results of the three algorithms we tested[12]. Second, it can support ψ_H.

4.3 Comparison Test II

In this test, we compared four previously proposed machine transliteration methods (Kang & Choi, 2000; Kang & Kim, 2000; Goto et al., 2003; Bilac & Tanaka, 2004) to the four transliteration models (ψ_G, ψ_P, ψ_H, and ψ_C), which were based on the MEM. Table 9 shows the results. We trained and tested the previous methods with the same data sets used for the four transliteration models. Table 10 shows the key features of the methods and models from the viewpoint of information type and usage. Information type indicates the type of information considered: source grapheme, source phoneme, and correspondence between the two. For example, the first three methods use only the source grapheme. Information usage indicates the context used and whether the previous output is used.

It is obvious from the table that the more information types a transliteration model considers, the better its performance. Either the source phoneme or the correspondence – which are not considered in the methods of Kang and Choi (2000), Kang and Kim (2000), and Goto *et al.* (2003) – is the key to the higher performance of the method of Bilac and Tanaka (2004) and the ψ_H and ψ_C.

From the viewpoint of information usage, the models and methods that consider the previous output tended to achieve better performance. For example, the method of Goto *et al.* (2003) had better results than that of Kang and Choi (2000). Because machine translit-

12. A one-tail paired t-test showed that the results with the MEM were always significantly better (except for ϕ_G in EJSet) than those of DTL and MBL (level of significance = 0.001).

Method/Model		EKSet	EJSet
Previous methods	Kang and Choi (2000)	51.4%	50.3%
	Kang and Kim (2000)	55.1%	53.2%
	Goto *et al.* (2003)	55.9%	56.2%
	Bilac and Tanaka (2004)	58.3%	62.5%
MEM-based models	ψ_G	58.8%	58.8%
	ψ_P	55.2%	59.2%
	ψ_H	64.1%	65.8%
	ψ_C	65.5%	69.1%

Table 9: Results of Comparison Test II.

Method/Model	Info. type			Info. usage	
	S	P	C	Context	PO
Kang and Choi (2000)	+	−	−	$< -3 \sim +3 >$	−
Kang and Kim (2000)	+	−	−	Unbounded	+
Goto *et al.* (2003)	+	−	−	$< -3 \sim +3 >$	+
Bilac and Tanaka (2004)	+	+	−	Unbounded	−
ψ_G	+	−	−	$< -3 \sim +3 >$	+
ψ_P	−	+	−	$< -3 \sim +3 >$	+
ψ_H	+	+	−	$< -3 \sim +3 >$	+
ψ_C	+	+	+	$< -3 \sim +3 >$	+

Table 10: Information type and usage for previous methods and four transliteration models, where S, P, C, and PO respectively represent the source grapheme, source phoneme, correspondence between S and P, and previous output.

eration is sensitive to context, a reasonable context size usually enhances transliteration ability. Note that the size of the context window for the previous methods was limited to 3 because a context window wider than 3 degrades performance (Kang & Choi, 2000) or does not significantly improve it (Kang & Kim, 2000). Experimental results related to context window size are given in Section 4.5.

Overall, ψ_H and ψ_C had better performance than the previous methods (on average, 17.04% better for EKSet and 21.78% better for EJSet), ψ_G (on average, 9.6% better for EKSet and 14.4% better for EJSet), and ψ_P (on average, 16.7% better for EKSet and 19.0% better for EJSet). In short, a good machine transliteration model should 1) consider either the correspondence between the source grapheme and the source phoneme or both the source grapheme and the source phoneme, 2) have a reasonable context size, and 3) consider previous output. The ψ_H and ψ_C satisfy all three conditions.

4.4 Dictionary Test

Table 11 shows the performance of each transliteration model for the dictionary test. In this test, we evaluated four transliteration models according to a way of pronunciation generation (or grapheme-to-phoneme conversion). *Registered* represents the performance for words registered in the pronunciation dictionary, and *Unregistered* represents that for unregistered words. On average, the number of *Registered* words in EKSet was about 600, and that in EJSet was about 700 in k-fold cross-validation test data. In other words, *Registered* words accounted for about 60% of the test data in EKSet and about 70% of the test data in EJSet. The correct pronunciation can always be acquired from the pronunciation dictionary for *Registered* words, while the pronunciation must be estimated for *Unregistered* words through automatic grapheme-to-phoneme conversion. However, the automatic grapheme-to-phoneme conversion does not always produce correct pronunciations – the estimated rate of correct pronunciations was about 70% accuracy.

	EKSet		EJSet	
	Registered	*Unregistered*	*Registered*	*Unregistered*
ψ_G	60.91%	55.74%	61.18%	50.24%
ψ_P	66.70%	38.45%	64.35%	40.78%
ψ_H	70.34%	53.31%	70.20%	50.02%
ψ_C	73.32%	54.12%	74.04%	51.39%
ALL	80.78%	68.41%	81.17%	62.31%

Table 11: Results of Dictionary Test: ALL means $\psi_{G+P+H+C}$.

Analysis of the results showed that the four transliteration models fall into three categories. Since the ψ_G is free from the need for correct pronunciation, that is, it does not use the source phoneme, its performance is not affected by pronunciation correctness. Therefore, ψ_G can be regarded as the baseline performance for *Registered* and *Unregistered*. Because ψ_P ($\phi_{PT} \circ \phi_{SP}$), ψ_H ($\alpha\times$ $Pr(\psi_P)$+$(1-\alpha)\times$ $Pr(\psi_G)$), and ψ_C ($\phi_{(SP)T} \circ \phi_{SP}$) depend on the source phoneme, their performance tends to be affected by the performance of ϕ_{SP}. Therefore, ψ_P, ψ_H, and ψ_C show notable differences in performance between *Registered* and *Unregistered*. However, the performance gap differs with the strength of the dependence. ψ_P falls into the second category: its performance strongly depends on the correct pronunciation. ψ_P tends to perform well for *Registered* and poorly for *Unregistered*. ψ_H and ψ_C weakly depend on the correct pronunciation. Unlike ψ_P, they make use of both the source grapheme and source phoneme. Therefore, they can perform reasonably well without the correct pronunciation because using the source grapheme weakens the negative effect of incorrect pronunciation in machine transliteration.

Comparing ψ_C and ψ_P, we find two interesting things. First, ψ_P was more sensitive to errors in ϕ_{SP} for *Unregistered*. Second, ψ_C showed better results for both *Registered* and *Unregistered*. Because ψ_P and ψ_C share the same function, ϕ_{SP}, the key factor accounting for the performance gap between them is the component functions, ϕ_{PT} and $\phi_{(SP)T}$. From the results shown in Table 11, we can infer that $\phi_{(SP)T}$ (in ψ_C) performed better than ϕ_{PT} (in ψ_P) for both *Registered* and *Unregistered*. In $\phi_{(SP)T}$, the source grapheme corresponding to the source phonemes, which ϕ_{PT} does not consider, made two contributions to the higher performance of $\phi_{(SP)T}$. First, the source grapheme in the correspondence made it possible to produce more accurate transliterations. Because $\phi_{(SP)T}$ considers the correspondence, $\phi_{(SP)T}$ has a more powerful transliteration ability than ϕ_{PT}, which uses just the source phonemes, when the correspondence is needed to produce correct transliterations. This is the main reason $\phi_{(SP)T}$ performed better than ϕ_{PT} for *Registered*. Second, source graphemes in the correspondence compensated for errors produced by ϕ_{SP} in producing target graphemes. This is the main reason $\phi_{(SP)T}$ performed better than ϕ_{PT} for *Unregistered*. In the comparison between ψ_C and ψ_G, the performances were similar for *Unregistered*. This indicates that the transliteration power of ψ_C is similar to that of ψ_G, even though the pronunciation of the source language word may not be correct. Furthermore, the performance of ψ_C was significantly higher than that of ψ_G for *Registered*. This indicates that the transliteration power of ψ_C is greater than that of ψ_G if the correct pronunciation is given.

The behavior of ψ_H was similar to that of ψ_C. For *Unregistered*, $Pr(\psi_G)$ in ψ_H made it possible for ψ_H to avoid errors caused by $Pr(\psi_P)$. Therefore, it worked better than ψ_P. For *Registered*, $Pr(\psi_P)$ enabled ψ_H to perform better than ψ_G.

The results of this test showed that ψ_H and ψ_C perform better than ψ_G and ψ_P while complementing ψ_G and ψ_P (and thus overcoming their disadvantage) by considering either the correspondence between the source grapheme and the source phoneme or both the source grapheme and the source phoneme.

4.5 Context Window-Size Test

In our testing of the effect of the context window size, we varied the size from 1 to 5. Regardless of the size, ψ_H and ψ_C always performed better than both ψ_G and ψ_P. When the size was 4 or 5, each model had difficulty identifying regularities in the training data. Thus, there were consistent drops in performance for all models when the size was increased from 3 to 4 or 5. Although the best performance was obtained when the size was 3, as shown in Table 12, the differences in performance were not significant in the range of 2-4. However, there was a significant difference between a size of 1 and a size of 2. This indicates that a lack of contextual information can easily lead to incorrect transliteration. For example, to produce the correct target language grapheme of *t* in *-tion*, we need the right three graphemes (or at least the right two) of *t*, *-ion* (or *-io*). The results of this testing indicate that the context size should be more than 1 to avoid degraded performance.

4.6 Training Data-Size Test

Table 13 shows the results of the Training Data-Size Test using MEM-based machine transliteration models. We evaluated the performance of the four models and *ALL* while varying the size of the training data from 20% to 100%. Obviously, the more training data used, the higher the system performance. However, the objective of this test was to determine whether the transliteration models perform reasonably well even for a small amount of training data. We found that ψ_G was the most sensitive of the four models to the amount of training data; it had the largest difference in performance between 20% and 100%. In contrast, *ALL* showed the smallest performance gap. The results of this test shows that

EKSet					
Context Size	ψ_G	ψ_P	ψ_H	ψ_C	*ALL*
1	44.9%	44.9%	51.8%	52.4%	65.8%
2	57.3%	52.8%	61.7%	64.4%	74.4%
3	58.8%	55.2%	64.1%	65.5%	75.8%
4	56.1%	54.6%	61.8%	64.3%	74.4%
5	53.7%	52.6%	60.4%	62.5%	73.9%
EJSet					
Context Size	ψ_G	ψ_P	ψ_H	ψ_C	*ALL*
1	46.4%	52.1%	58.0%	62.0%	70.4%
2	58.2%	59.5%	65.6%	68.7%	76.3%
3	58.8%	59.2%	65.8%	69.1%	77.0%
4	56.4%	58.5%	64.4%	68.2%	76.0%
5	53.9%	56.4%	62.9%	66.3%	75.5%

Table 12: Results of Context Window-Size Test: ALL means $\psi_{G+P+H+C}$.

combining different transliteration models is helpful in producing correct transliterations even if there is little training data.

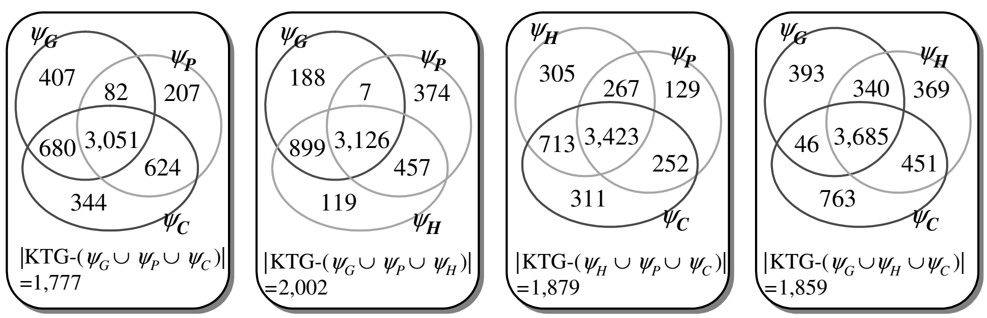
EKSet					
Training Data Size	ψ_G	ψ_P	ψ_H	ψ_C	*ALL*
20%	46.6%	47.3%	53.4%	57.0%	67.5%
40%	52.6%	51.5%	58.7%	62.1%	71.6%
60%	55.2%	53.0%	61.5%	63.3%	73.0%
80%	58.9%	54.0%	62.6%	64.6%	74.7%
100%	58.8%	55.2%	64.1%	65.5%	75.8%
EJSet					
Training Data Size	ψ_G	ψ_P	ψ_H	ψ_C	*ALL*
20%	47.6%	51.2%	56.4%	60.4%	69.6%
40%	52.4%	55.1%	60.7%	64.8%	72.6%
60%	55.2%	57.3%	62.9%	66.6%	74.7%
80%	57.9%	58.8%	65.4%	68.0%	76.7%
100%	58.8%	59.2%	65.8%	69.1%	77.0%

Table 13: Results of Training Data-Size Test: ALL means $\psi_{G+P+H+C}$.

5. Discussion

Figures 3 and 4 show the distribution of the correct transliterations produced by each transliteration model and by the combination of models, all based on the MEM. The ψ_G,

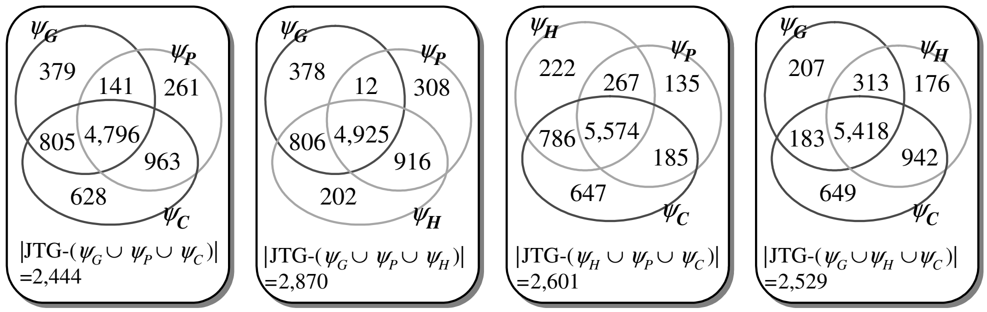
ψ_P, ψ_H, and ψ_C in the figures represent the set of correct transliterations produced by each model through k-fold validation. For example, $|\psi_G|$ = 4,220 for EKSet and $|\psi_G|$ = 6,121 for EJSet mean that ψ_G produced 4,220 correct transliterations for 7,172 English words in EKSet ($|KTG|$ in Figure 3) and 6,121 correct ones for 10,417 English words in EJSet ($|JTG|$ in Figure 4). An important factor in modeling a transliteration model is to reflect the dynamic transliteration behaviors, which means that a transliteration process dynamically uses the source grapheme and source phoneme to produce transliterations. Due to these dynamic behaviors, a transliteration can be grapheme-based transliteration, phoneme-based transliteration, or some combination of the two. The forms of transliterations are classified on the basis of the information upon which the transliteration process mainly relies (either a source grapheme or a source phoneme or some combination of the two). Therefore, an effective transliteration system should be able to produce various types of transliterations at the same time. One way to accommodate the different dynamic transliteration behaviors is to combine different transliteration models, each of which can handle a different behavior. Synergy can be achieved by combining models so that one model can produce the correct transliteration when the others cannot. Naturally, if the models tend to produce the same transliteration, less synergy can be realized from combining them. Figures 3 and 4 show the synergy gained from combining transliteration models in terms of the size of the intersection and the union of the transliteration models.



(a) ψ_G+ψ_P+ψ_C (b) ψ_G+ψ_P+ψ_H (c) ψ_P+ψ_H+ψ_C (d) ψ_G+ψ_H+ψ_C

Figure 3: Distributions of correct transliterations produced by models for English-to-Korean transliteration. KTG represents "Korean Transliterations in the Gold standard". Note that $|\psi_G \cup \psi_P \cup \psi_H \cup \psi_C|$ = 5,439, $|\psi_G \cap \psi_P \cap \psi_H \cap \psi_C|$ = 3,047, and $|KTG|$ = 7,172.

The figures show that, as the area of intersection between different transliteration models becomes smaller, the size of their union tends to become bigger. The main characteristics obtained from these figures are summarized in Table 14. The first thing to note is that $|\psi_G \cap \psi_P|$ is clearly smaller than any other intersection. The main reason for this is that ψ_G and ψ_P use no common information (ψ_G uses source graphemes while ψ_P uses source phonemes). However, the others use at least one of source grapheme and source phoneme (source graphemes are information common to ψ_G, ψ_H, and ψ_C while source phonemes are information common to ψ_P, ψ_H, and ψ_C). Therefore, we can infer that the synergy derived from combining ψ_G and ψ_P is greater than that derived from the other combinations.



(a) ψ_G+ψ_P+ψ_C (b) ψ_G+ψ_P+ψ_H (c) ψ_P+ψ_H+ψ_C (d) ψ_G+ψ_H+ψ_C

Figure 4: Distributions of correct transliterations produced by models for English-to-Japanese transliteration. JTG represents "Japanese Transliterations in the Gold standard". Note that $|\psi_G \cup \psi_P \cup \psi_H \cup \psi_C|$=8,021, $|\psi_G \cap \psi_P \cap \psi_H \cap \psi_C|$=4,786, and $|JTG|$ = 10,417.

	EKSet	EJSet
$\lvert\psi_G\rvert$	4,202	6,118
$\lvert\psi_P\rvert$	3,947	6,158
$\lvert\psi_H\rvert$	4,583	6,846
$\lvert\psi_C\rvert$	4,680	7,189
$\lvert\psi_G \cap \psi_P\rvert$	3,133	4,937
$\lvert\psi_G \cap \psi_C\rvert$	3,731	5,601
$\lvert\psi_G \cap \psi_H\rvert$	4,025	5,731
$\lvert\psi_C \cap \psi_H\rvert$	4,136	6,360
$\lvert\psi_P \cap \psi_C\rvert$	3,675	5,759
$\lvert\psi_P \cap \psi_H\rvert$	3,583	5,841
$\lvert\psi_G \cup \psi_P\rvert$	5,051	7,345
$\lvert\psi_G \cup \psi_C\rvert$	5,188	7,712
$\lvert\psi_G \cup \psi_H\rvert$	4,796	7,239
$\lvert\psi_C \cup \psi_H\rvert$	5,164	7,681
$\lvert\psi_P \cup \psi_C\rvert$	4,988	7,594
$\lvert\psi_P \cup \psi_H\rvert$	4,982	7,169

Table 14: Main characteristics obtained from Figures 3 and 4.

However, the size of the union between the various pairs of transliteration models in Table 14 shows that $|\psi_C \cup \psi_H|$ and $|\psi_G \cup \psi_C|$ are bigger than $|\psi_G \cup \psi_P|$. The main reason for this might be the higher transliteration power of ψ_C and ψ_H compared to that of ψ_G and ψ_P – ψ_C and ψ_H cover more of the KTG and JTG than both ψ_G and ψ_P. The second thing to note is that the contribution of each transliteration model to $|\psi_G \cup \psi_P \cup \psi_H \cup \psi_C|$ can be estimated from the difference between $|\psi_G \cup \psi_P \cup \psi_H \cup \psi_C|$ and the union of the three other transliteration models. For example, we can measure the contribution of ψ_H from the difference between $|\psi_G \cup \psi_P \cup \psi_H \cup \psi_C|$ and $|\psi_G \cup \psi_P \cup \psi_C|$. As shown in Figures 3(a) and 4(a)), ψ_H makes the smallest contribution while ψ_C (Figures 3(b) and 4(b)) makes the largest contribution. The main reason for ψ_H making the smallest contribution is that it tends to produce the same transliteration as the others, so the intersection between ψ_H and the others tends to be large.

It is also important to rank the transliterations produced by a transliteration system for a source language word on the basis of their relevance. While a transliteration system can produce a list of transliterations, each reflecting a dynamic transliteration behavior, it will fail to perform well unless it can distinguish between correct and wrong transliterations. Therefore, a transliteration system should be able to produce various kinds of transliterations depending on the dynamic transliteration behaviors and be able to rank them on the basis of their relevance. In addition, the application of transliteration results to natural language applications such as machine translation and information retrieval requires that the transliterations be ranked and assigned a relevance score.

In summary, 1) **producing a list of transliterations reflecting dynamic transliteration behaviors** (one way is to combine the results of different transliteration models, each reflecting one of the dynamic transliteration behaviors) and 2) **ranking the transliterations in terms of their relevance** are both necessary to improve the performance of machine transliteration. In the next section, we describe a way to calculate the relevance of transliterations produced by a combination of the four transliteration models.

6. Transliteration Ranking

The basic assumption of transliteration ranking is that correct transliterations are more frequently used in real-world texts than incorrect ones. Web data reflecting the real-world usage of transliterations can thus be used as a language resource to rank transliterations. Transliterations that appear more frequently in web documents are given either a higher rank or a higher score. The goal of transliteration ranking, therefore, is to rank correct transliterations higher and rank incorrect ones lower. The transliterations produced for a given English word by the four transliteration models (ψ_G, ψ_P, ψ_H, and ψ_C), based on the MEM, were ranked using web data.

Our transliteration ranking relies on web frequency (number of web documents). To obtain reliable web frequencies, it is important to consider a transliteration and its corresponding source language word together rather than the transliteration alone. This is because our aim is to find correct transliterations corresponding to a source language word rather than to find transliterations that are frequently used in the target language. Therefore, the best approach to transliteration ranking using web data is to find web documents in which transliterations are used as translations of the source language word.

A bilingual phrasal search (BPS) retrieves the Web with a Web search engine query, which is a phrase composed of a transliteration and its source language word (e.g., {'a-mil-la-a-je' *amylase*}). The BPS enables the Web search engine to find web documents that contain correct transliterations corresponding to the source language word. Note that a phrasal query is represented in brackets, where the first part is a transliteration and the second part is the corresponding source language word. Figure 5 shows Korean and Japanese web documents retrieved using a BPS for *amylase* and its Korean/Japanese transliterations,

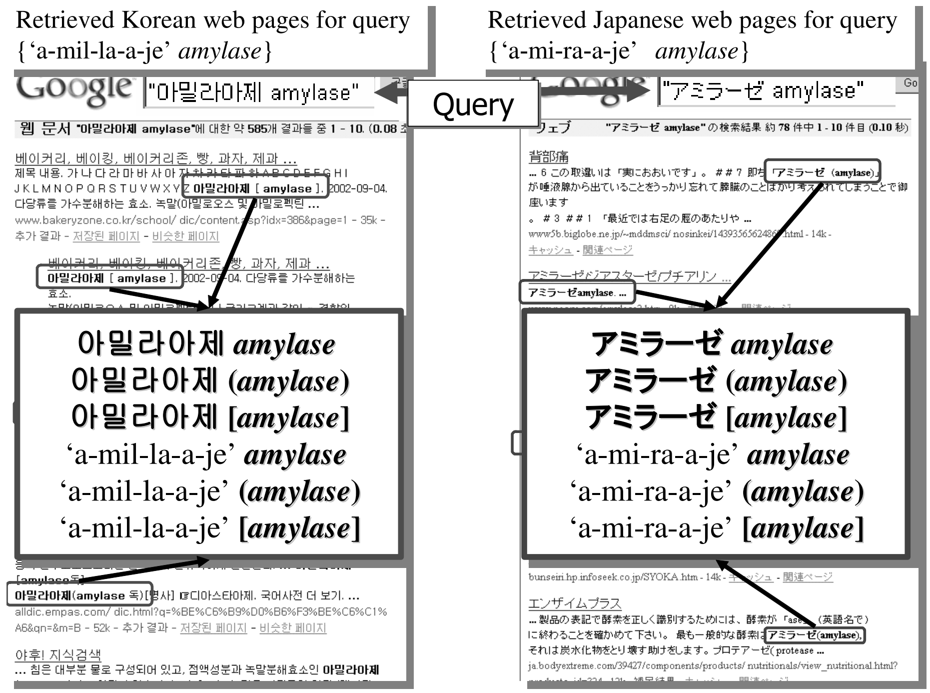


Figure 5: Desirable retrieved web pages for transliteration ranking.

'a-mil-la-a-je' and 'a-mi-ra-a-je'. The web documents retrieved by a BPS usually contain a transliteration and its corresponding source language word as a translation pair, with one of them often placed in parentheses, as shown in Figure 5.

A dilemma arises, though, regarding the quality and coverage of retrieved web documents. Though a BPS generally provides high-quality web documents that contain correct transliterations corresponding to the source language word, the coverage is relatively low, meaning that it may not find any web documents for some transliterations. For example, a BPS for the Japanese phrasal query {'a-ru-ka-ro-si-su' *alkalosis*} and the Korean phrasal query {'eo-min' *ermine*} found no web documents. Therefore, alternative search methods are necessary when the BPS fails to find any relevant web documents. A bilingual keyword search (BKS) (Qu & Grefenstette, 2004; Huang, Zhang, & Vogel, 2005; Zhang, Huang, & Vogel, 2005) can be used when the BPS fails, and a monolingual keyword search (MKS) (Grefenstette, Qu, & Evans, 2004) can be used when both the BPS and BKS fail. Like a BPS, a BKS makes use of two keywords, a transliteration and its source language word, as a search engine query. Whereas a BPS retrieves web documents containing the two keywords as a phrase, a BKS retrieves web documents containing them anywhere in the document. This means that the web frequencies of noisy transliterations are sometimes higher than those of correct transliterations in a BKS, especially when the noisy transliterations are one-syllable transliterations. For example, 'mok', which is a Korean transliteration produced for *mook* and a one-syllable noisy transliteration, has a higher web frequency than 'mu-keu', which is the correct transliteration for *mook*, because 'mok' is a common Korean noun that frequently appears in Korean texts with the meaning of *neck*. However, a BKS can improve coverage without a great loss of quality in the retrieved web documents if the transliterations are composed of two or more syllables.

Though a BKS has higher coverage than a BPS, it can fail to retrieve web documents in some cases. In such cases, an MKS (Grefenstette et al., 2004) is used. In an MKS, a transliteration alone is used as the search engine query. A BPS and a BKS act like a translation model, while an MKS acts like a language model. Though an MKS cannot give information as to whether the transliteration is correct, it does provide information as to whether the transliteration is likely to be a target language word. The three search methods are used sequentially (BPS, BKS, MKS). If one method fails to retrieve any relevant web documents, the next one is used. Table 15 summarizes the conditions for applying each search method.

Along with these three search strategies, three different search engines are used to obtain more web documents. The search engines used for this purpose should satisfy two conditions: 1) support Korean/Japanese web document retrieval and 2) support both phrasal and keyword searches. Google[13], Yahoo[14], and MSN[15] satisfy these conditions, and we used them as our search engines.

Search method	Condition
BPS	$\sum_j \sum_{c_k \in C} WF_{BPSj}(e, c_k)) \neq 0$
BKS	$\sum_j \sum_{c_k \in C} WF_{BPSj}(e, c_k)) = 0$ $\sum_j \sum_{c_k \in C} WF_{BKSj}(e, c_k)) \neq 0$
MKS	$\sum_j \sum_{c_k \in C} WF_{BPSj}(e, c_k) = 0$ $\sum_j \sum_{c_k \in C} WF_{BKSj}(e, c_k) = 0$ $\sum_j \sum_{c_k \in C} WF_{MKSj}(e, c_k) \neq 0$

Table 15: Conditions under which each search method is applied.

$$RF(e, c_i) = \sum_j NWF_j(e, c_i) = \sum_j \frac{WF_j(e, c_i)}{\sum_{c_k \in C} WF_j(e, c_k)} \tag{7}$$

Web frequencies acquired from these three search methods and these three search engines were used to rank transliterations on the basis of Formula (7), where c_i is the i^{th} transliteration produced by the four transliteration models, e is the source language word of c_i, RF is a function for ranking transliterations, WF is a function for calculating web frequency, NWF is a function for normalizing web frequency, C is a set of produced transliterations, and j is an index for the j^{th} search engine. We used the normalized web frequency as a ranking factor. The normalized web frequency is the web frequency divided by the total web frequency of all produced transliterations corresponding to one source language word. The score for a transliteration is then calculated by summing up the normalized

13. `http://www.google.com`
14. `http://www.yahoo.com`
15. `http://www.msn.com`

web frequencies of the transliteration given by the three search engines. Table 16 shows an example ranking for the English word *data* and its possible Korean transliterations, 'de-i-teo', 'de-i-ta', and 'de-ta', which web frequencies are obtained using a BPS. The normalized WF_{BPS} (NWF_{BPS}) for search engine A was calculated as follows.

- NWF_{BPS} (*data*, 'de-i-teo') = 94,100 / (94,100 + 67,800 + 54) = 0.5811
- NWF_{BPS} (*data*, 'de-i-ta') = 67,800 / (94,100 + 67,800 + 54) = 0.4186
- NWF_{BPS} (*data*, 'de-ta') = 54 / (94,100 + 67,800 + 54) = 0.0003

The ranking score for 'de-i-teo' was then calculated by summing up NWF_{BPS} (*data*, 'de-i-teo') for each search engine:

- RF_{BPS} (*data*, 'de-i-teo') = 0.5810 + 0.7957 + 0.3080 = 1.6848

Search Engine	e=*data*					
	c_1= 'de-i-teo'		c_2= 'de-i-ta'		c_3= 'de-ta'	
	WF	NWF	WF	NWF	WF	NWF
A	94,100	0.5811	67,800	0.4186	54	0.0003
B	101,834	0.7957	26,132	0.2042	11	0.0001
C	1,358	0.3080	3,028	0.6868	23	0.0052
RF	1.6848		1.3096		0.0056	

Table 16: Example transliteration ranking for *data* and its transliterations; WF, NWF, and RF represent WF_{BPS}, NWF_{BPS}, and RF_{BPS}, respectively.

6.1 Evaluation

We tested the performance of the transliteration ranking under two conditions: 1) with all test data (ALL) and 2) with test data for which at least one transliteration model produced the correct transliteration (CTC). Testing with ALL showed the overall performance of the machine transliteration while testing with CTC showed the performance of the transliteration ranking alone. We used the performance of the individual transliteration models (ψ_G, ψ_P, ψ_H, and ψ_C) as the baseline. The results are shown in Table 17. "Top-n" means that the correct transliteration was within the Top-n ranked transliterations. The average number of produced Korean transliterations was 3.87 and that of Japanese ones was 4.50; note that ψ_P and ψ_C produced more than one transliteration because of pronunciation variations. The results for both English-to-Korean and English-to-Japanese transliteration indicate that our ranking method effectively filters out noisy transliterations and positions the correct transliterations in the top rank; most of the correct transliterations were in Top-1. We see that transliteration ranking (in Top-1) significantly improved performance of the individual models for both EKSet and EJSet[16]. The overall performance of the

16. A one-tail paired t-test showed that the performance improvement was significant (level of significance = 0.001.

transliteration (for ALL) as well that of the ranking (for CTC) were relatively good. Notably, the CTC performance showed that web data is a useful language resource for ranking transliterations.

Test data		EKSet	EJSet
ALL	ψ_G	58.8%	58.8%
	ψ_P	55.2%	59.2%
	ψ_H	64.1%	65.8%
	ψ_C	65.5%	69.1%
ALL	Top-1	71.5%	74.8%
	Top-2	75.3%	76.9%
	Top-3	75.8%	77.0%
CTC	Top-1	94.3%	97.2%
	Top-2	99.2%	99.9%
	Top-3	100%	100%

Table 17: Results of Transliteration ranking.

6.2 Analysis of Results

We defined two error types: **production errors** and **ranking errors**. A production error is when there is no correct transliteration among the produced transliterations. A ranking error is when the correct transliteration does not appear in the Top-1 ranked transliterations.

We examined the relationship between the search method and the transliteration ranking. Table 18 shows the ranking performance by each search method. The RTC represents correct transliterations ranked by each search method. The NTC represents test data ranked, that is, the coverage of each search method. The ratio of RTC to NTC represents the upper bound of performance and the difference between RTC and NTC is the number of errors.

The best performance was with a BPS. A BPS handled 5,270 out of 7,172 cases for EKSet and 8,829 out of 10,417 cases for EJSet. That is, it did the best job of retrieving web documents containing transliteration pairs. Analysis of the ranking errors revealed that the main cause of such errors in a BPS was transliteration variations. These variations contribute to ranking errors in two ways. First, when the web frequencies of transliteration variations are higher than those of the standard ones, the variations are ranked higher than the standard ones, as shown by the examples in Table 19. Second, when the transliterations include only transliteration variations (i.e., there are no correct transliterations), the correct ranking cannot be. In this case, ranking errors are caused by production errors. With a BPS, there were 603 cases of this for EKSet and 895 cases for EJSet.

NTC was smaller with a BKS and an MKS because a BPS retrieves web documents whenever possible. Table 18 shows that production errors are the main reason a BPS fails to retrieve web documents. (When a BKS or MKS was used, production errors occurred in

	EKSet			EJSet		
	BPS	BKS	MKS	BPS	BKS	MKS
Top-1	83.8%	55.1%	16.7%	86.2%	19.0%	2.7%
Top-2	86.6%	58.4%	27.0%	88.3%	22.8%	4.2%
Top-3	86.6%	58.2%	31.3%	88.35%	22.9%	4.3%
RTC	4,568	596	275	7,800	188	33
NTC	5,270	1,024	878	8,829	820	768

Table 18: Ranking performance of each search method.

	Transliteration	Web Frequency
compact → Korean	‘kom-paek-teu’	1,075
	‘keom-paek-teu’∗	1,793
pathos → Korean	‘pa-to-seu’	1,615
	‘pae-to-seu’∗	14,062
cohen → Japanese	‘ko-o-he-n’	23
	‘ko-o-e-n’∗	112
criteria → Japanese	‘ku-ra-i-te-ri-a’	104
	‘ku-ri-te-ri-a’∗	1,050

Table 19: Example ranking errors when a BPS was used (∗ indicates a variation).

all but 871[17] cases for EKSet and 221[18] cases for EJSet). The results also show that a BKS was more effective than an MKS.

The trade-off between the quality and coverage of retrieved web documents is an important factor in transliteration ranking. A BPS provides better quality rather than wider coverage, but is effective since it provides reasonable coverage.

7. Conclusion

We tested and compared four transliteration models, **grapheme-based transliteration model** (ψ_G), **phoneme-based transliteration model** (ψ_P), **hybrid transliteration model** (ψ_H), and **correspondence-based transliteration model** (ψ_C), for English-to-Korean and English-to-Japanese transliteration. We modeled a framework for the four transliteration models and compared them within the framework. Using the results, we examined a way to improve the performance of machine transliteration.

We found that the ψ_H and ψ_C are more effective than the ψ_G and ψ_P. The main reason for the better performance of ψ_C is that it uses the correspondence between the source grapheme and the source phoneme. The use of this correspondence positively affected transliteration performance in various tests.

17. 596 (RTC of BKS in EKSet) + 275 (RTC of MKS in EKSet) = 871
18. 188 (RTC of BKS for EJSet) + 33 (RTC of MKS for EJSet) = 221

We demonstrated that ψ_G, ψ_P, ψ_H, and ψ_C can be used as complementary transliteration models to improve the chances of producing correct transliterations. A combination of the four models produced more correct transliterations both in English-to-Korean transliteration and English-to-Japanese transliteration compared to each model alone. Given these results, we described a way to improve machine transliteration that combines different transliteration models: 1) **produce a list of transliterations by combining transliterations produced by multiple transliteration models**; 2) **rank the transliterations on the basis of their relevance**.

Testing showed that transliteration ranking based on web frequency is an effective way to calculate the relevance of transliterations. This is because web data reflects real-world usage, so it can be used to filter out noisy transliterations, which are not used as target language words or are incorrect transliterations for a source language word.

There are several directions for future work. Although we considered some transliteration variations, our test sets mainly covered standard transliterations. In corpora or web pages, however, we routinely find other types of transliterations – misspelled transliterations, transliterations of common phrases, etc. – along with the standard transliterations and transliteration variations. Therefore, further testing using such transliterations is needed to enable the transliteration models to be compared more precisely. To achieve a machine transliteration system capable of higher performance, we need a more sophisticated transliteration method and a more sophisticated ranking algorithm. Though many correct transliterations can be acquired through the combination of the four transliteration models, there are still some transliterations that none of the models can produce. We need to devise a method that can produce them. Our transliteration ranking method works well, but, because it depends on web data, it faces limitations if the correct transliteration does not appear in web data. We need a complementary ranking method to handle such cases. Moreover, to demonstrate the effectiveness of these four transliteration models, we need to apply them to various natural language processing applications.

Acknowledgments

We are grateful to Claire Cardie and the anonymous reviewers for providing constructive and insightful comments to earlier drafts of this paper.

References

Aha, D. W. (1997). Lazy learning: Special issue editorial. *Artificial Intelligence Review, 11:710*.

Aha, D. W., Kibler, D., & Albert, M. (1991). Instance-based learning algorithms. *Machine Learning*, *6*(3766).

Al-Onaizan, Y., & Knight, K. (2002). Translating named entities using monolingual and bilingual resources. In *Proceedings of ACL 2002*, pp. 400–408.

Andersen, O., Kuhn, R., Lazarides, A., Dalsgaard, P., Haas, J., & Noth, E. (1996). Comparison of two tree-structured approaches for grapheme-to-phoneme conversion. In *Proceedings of ICSLP 1996*, pp. 1808–1811.

Berger, A. L., Pietra, S. D., & Pietra, V. J. D. (1996). A maximum entropy approach to natural language processing. *Computational Linguistics*, *22*(1), 39–71.

Bilac, S., & Tanaka, H. (2004). Improving back-transliteration by combining information sources. In *Proceedings of IJCNLP2004*, pp. 542–547.

Breen, J. (2003). EDICT Japanese/English dictionary .le. The Electronic Dictionary Research and Development Group, Monash University. `http://www.csse.monash.edu.au/~jwb/edict.html`.

Chen, S. F. (2003). Conditional and joint models for grapheme-to-phoneme conversion. In *Proceedings of Eurospeech*, pp. 2033–2036.

CMU (1997). The CMU pronouncing dictionary version 0.6. `http://www.speech.cs.cmu.edu/cgi-bin/cmudict`.

Cover, T. M., & Hart, P. E. (1967). Nearest neighbor pattern classification. *Institute of Electrical and Electronics Engineers Transactions on Information Theory*, *13*(2127).

Daelemans, W., Zavrel, J., Sloot, K. V. D., & Bosch, A. V. D. (2004). TiMBL: Tilburg Memory-Based Learner - version 5.1 reference guide. Tech. rep. 04-02, ILK Technical Report Series.

Daelemans, W., & van den Bosch, A. (1996). Language-independent data-oriented grapheme-to-phoneme conversion. In J. Van Santen, R. Sproat, J. O., & Hirschberg, J. (Eds.), *Progress in Speech Synthesis*, pp. 77–90. Springer Verlag, New York.

Damper, R. I., Marchand, Y., Adamson, M. J., & Gustafson, K. (1999). Evaluating the pronunciation component of text-to-speech systems for English: A performance comparison of different approaches. *Computer Speech and Language*, *13*(2), 155–176.

Devijver, P. A., & Kittler., J. (1982). *Pattern recognition: A statistical approach*. Prentice-Hall.

Fujii, A., & Tetsuya, I. (2001). Japanese/English cross-language information retrieval: Exploration of query translation and transliteration. *Computers and the Humanities*, *35*(4), 389–420.

Goto, I., Kato, N., Uratani, N., & Ehara, T. (2003). Transliteration considering context information based on the maximum entropy method. In *Proceedings of MT-Summit IX*, pp. 125–132.

Grefenstette, G., Qu, Y., & Evans, D. A. (2004). Mining the web to create a language model for mapping between English names and phrases and Japanese. In *Proceedings of Web Intelligence*, pp. 110–116.

Huang, F., Zhang, Y., & Vogel, S. (2005). Mining key phrase translations from web corpora. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pp. 483–490.

Jeong, K. S., Myaeng, S. H., Lee, J. S., & Choi, K. S. (1999). Automatic identification and back-transliteration of foreign words for information retrieval. *Information Processing and Management*, *35*(1), 523–540.

Jung, S. Y., Hong, S., & Paek, E. (2000). An English to Korean transliteration model of extended markov window. In *Proceedings of the 18th conference on Computational linguistics*, pp. 383 – 389.

Kang, B. J. (2001). *A resolution of word mismatch problem caused by foreign word transliterations and English words in Korean information retrieval.* Ph.D. thesis, Computer Science Dept., KAIST.

Kang, B. J., & Choi, K. S. (2000). Automatic transliteration and back-transliteration by decision tree learning. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation*, pp. 1135–1411.

Kang, I. H., & Kim, G. C. (2000). English-to-Korean transliteration using multiple unbounded overlapping phoneme chunks. In *Proceedings of the 18th International Conference on Computational Linguistics*, pp. 418–424.

Kim, J. J., Lee, J. S., & Choi, K. S. (1999). Pronunciation unit based automatic English-Korean transliteration model using neural network. In *Proceedings of Korea Cognitive Science Association*, pp. 247–252.

Knight, K., & Graehl, J. (1997). Machine transliteration. In *Proceedings of the 35th Annual Meetings of the Association for Computational Linguistics*, pp. 128–135.

Korea Ministry of Culture & Tourism (1995). English to Korean standard conversion rule. `http://www.hangeul.or.kr/nmf/23f.pdf`.

Lee, J. S. (1999). *An English-Korean transliteration and retransliteration model for Cross-lingual information retrieval.* Ph.D. thesis, Computer Science Dept., KAIST.

Lee, J. S., & Choi, K. S. (1998). English to Korean statistical transliteration for information retrieval. *Computer Processing of Oriental Languages*, *12*(1), 17–37.

Li, H., Zhang, M., & Su, J. (2004). A joint source-channel model for machine transliteration. In *Proceedings of ACL 2004*, pp. 160–167.

Lin, W. H., & Chen, H. H. (2002). Backward machine transliteration by learning phonetic similarity. In *Proceedings of the Sixth Conference on Natural Language Learning (CoNLL)*, pp. 139–145.

Manning, C., & Schutze, H. (1999). *Foundations of Statistical natural language Processing.* MIT Press.

Meng, H., Lo, W.-K., Chen, B., & Tang, K. (2001). Generating phonetic cognates to handle named entities in English-Chinese cross-language spoken document retrieval. In *Proceedings of Automatic Speech Recognition and Understanding, 2001. ASRU '01*, pp. 311–314.

Mitchell, T. M. (1997). *Machine learning.* New-York: McGraw-Hill.

Nam, Y. S. (1997). *Foreign dictionary.* Sung An Dang.

Oh, J. H., & Choi, K. S. (2002). An English-Korean transliteration model using pronunciation and contextual rules. In *Proceedings of COLING2002*, pp. 758–764.

Pagel, V., Lenzo, K., & Black, A. W. (1998). Letter to sound rules for accented lexicon compression. In *Proceedings of International Conference on Spoken Language Processing*, pp. 2015–2018.

Qu, Y., & Grefenstette, G. (2004). Finding ideographic representations of Japanese names written in Latin script via language identification and corpus validation.. In *Proc. of ACL*, pp. 183–190.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, *1*, 81–106.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kauffman.

Stalls, B. G., & Knight, K. (1998). Translating names and technical terms in arabic text. In *Proceedings of COLING/ACL Workshop on Computational Approaches to Semitic Languages*, pp. 34–41.

Zhang, L. (2004). Maximum entropy modeling toolkit for python and C++. `http://homepages.inf.ed.ac.uk/s0450736/software/maxent/manual.pdf`.

Zhang, Y., Huang, F., & Vogel, S. (2005). Mining translations of OOV terms from the web through cross-lingual query expansion. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 669–670.