ARTICLE IN PRESS

Available online at www.sciencedirect.com

SCIENCE DIRECT®

ELSEVIER

Information Processing and Management xxx (2005) xxx–xxx

INFORMATION PROCESSING & MANAGEMENT

www.elsevier.com/locate/infoproman

An ensemble of transliteration models for information retrieval

Jong-Hoon Oh *, Key-Sun Choi

Computer Science Division, Department of EECS, Korea Terminology Research Center for Language and Knowledge Engineering (KORTERM), Korea Advanced Institute of Science and Technology (KAIST), 373-1 Guseong-Dong, Yuseong-Gu, Daejeon 305-701, Republic of Korea

Received 21 June 2005; accepted 29 September 2005

Abstract

Transliteration is used to phonetically translate proper names and technical terms especially from languages in Roman alphabets to languages in non-Roman alphabets such as from English to Korean, Japanese, and Chinese. Because transliterations are usually representative index terms for documents, proper handling of the transliterations is important for an effective information retrieval system. However, there are limitations on handling transliterations depending on dictionary lookup, because transliterations are usually not registered in the dictionary. For this reason, many researchers have been trying to overcome the problem using machine transliteration. In this paper, we propose a method for improving machine transliteration using an ensemble of three different transliteration models. Because one transliteration model alone has limitation on reflecting all possible transliteration behaviors, several transliteration models should be complementary used in order to achieve a high-performance machine transliteration system. This paper describes a method about transliteration production using the several machine transliteration models and transliteration ranking with web data and relevance scores given by each transliteration model. We report evaluation results for our ensemble transliteration model and experimental results for its impact on IR effectiveness. Machine transliteration tests on English-to-Korean transliteration and English-to-Japanese transliteration show that our proposed method achieves 78–80% word accuracy. Information retrieval tests on KTSET and NTCIR-1 test collection show that our transliteration model can improve the performance of an information retrieval system about 10–34%.

Keywords: Machine transliteration; Ensemble-based transliteration model; Web data; Information retrieval; Machine learning

* Corresponding author. Address: National Institute of Information and Communications Technology, Information and Network Systems Department, Computational Linguistics Group, 3-5 Hikaridai Seika-cho, Soraku-gun, Kyoto, 619-0289 Japan. Tel.: +81 0774 98 6862; fax: +81 0774 98 6961.

E-mail addresses: rovellia@nict.go.jp, rovellia@world.kaist.ac.kr (J.-H. Oh), kschoi@world.kaist.ac.kr (K.-S. Choi).

0306-4573/$ - see front matter
doi:10.1016/j.ipm.2005.09.007

1. Introduction

In this paper, we propose method for improving machine transliteration using an ensemble of several different transliteration models. First, this paper describes a method about transliteration production using different machine transliteration models and transliteration ranking with web data and relevance scores given by each transliteration model in English-to-Korean transliteration and in English-to-Japanese transliteration. Web data as a context for weighting each produced transliteration filters out noisy transliterations. Finally, we present impact of our machine transliteration method on IR effectiveness through experiments.

Machine transliteration is an automatic method to generate characters or words in one alphabetical system for the corresponding characters in another alphabetical system. For example, the English word ''data'' is transliterated into Korean 'deita'[1] and Japanese 'deeta'. Transliteration is used to phonetically translate proper names and technical terms especially from languages in Roman alphabets to languages in non-Roman alphabets such as from English to Korean, Japanese, and Chinese. Because the proper names and technical terms, which are frequently transliterated, are usually selected as representative index terms for texts, proper handling of the transliterations is important for an effective information retrieval system. Transliterations are one of the main sources of the Out-of-Vocabulary (OOV) problem (Fujii & Ishikawa, 2001; Lin & Chen, 2002). Handling transliterations depending on dictionary lookup, therefore, is an impractical way. One way to solve the problem is machine transliteration. Nowadays, machine transliteration as an assistant of cross language applications, such as Machine Translation (MT) (Al-Onaizan & Knight, 2002; Knight & Graehl, 1997), and Cross-Language Information Retrieval (CLIR) (Fujii & Ishikawa, 2001; Lin & Chen, 2002), has been recognized as an important research issue. In the area of CLIR, machine transliteration bridges the gap between the transliterated localized form and its original form by generating all possible transliterated forms from their original form (or all possible original forms from their transliterated form). For example, machine transliteration gives a help to query translation in CLIR where proper names and technical terms frequently appear in source language queries. In the area of MT, machine transliteration prevents translation failure when translations of proper names and technical terms are not registered in a translation dictionary. Machine transliteration, therefore, may affect the performance of MT and CLIR system.

Three machine transliteration models have been studied as described in Fig. 1: called **grapheme[2]-based transliteration model** (ψ_G), **phoneme[3]-based transliteration model** (ψ_P), and **grapheme- and phoneme-based transliteration model** (ψ_{GP}). They are classified in terms of units to be transliterated. ψ_G is referred to the direct model because ψ_G directly transforms source language graphemes to target language graphemes without any phonetic knowledge of source language words (φ_{ST}). ψ_P is referred to the pivot model, because ψ_P makes use of phonemes as a pivot when it produces target language graphemes from source language graphemes. Therefore ψ_P usually needs two steps: the first step is to produce phonemes from source language graphemes (φ_{SP}), and the second step is to produce target language graphemes from phonemes (φ_{PT}).[4] ψ_{GP} makes use of both source language grapheme and phoneme ($\varphi_{(SP)T}$) when it produces target language transliterations. Hereafter, we will use a source grapheme for a source language grapheme and a target grapheme for a target language grapheme.

Transliterations produced by the three models are usually different because they depend on different information. Basically, transliteration is the phonetic process, like ψ_P, rather than the orthographic one, like ψ_G (Knight & Graehl, 1997). However, the standard transliterations in the real usage are not restricted to phoneme-based transliterations. For example, the standard Korean transliterations of ''data'', ''amylase'', and ''neomycin'' are the phoneme-based transliteration, 'deita', the grapheme-based transliteration, 'amillaaje',

[1] In this paper, target language transliterations are represented as their Romanization form in a quotation mark, like 'deita'.

[2] *Graphemes* refer to the basic units (or the smallest contrastive units) of written language: for example, English has 26 graphemes or letters, Korean has 24, Japanese has 50, and German has 30.

[3] *Phonemes* are the simplest significant unit of sound (or the smallest contrastive units of the spoken language): for example, the /M/, /AE/, and /TH/ in *math*.

[4] Depending on implementation, the two steps in the phoneme-based transliteration model are either explicit if a transliteration system produces target language transliterations after producing pronunciation of source language words or implicit if a transliteration system implicitly uses phonemes in the transliteration stage but it explicitly uses them in the learning stage like Bilac and Tanaka's (2004) method.

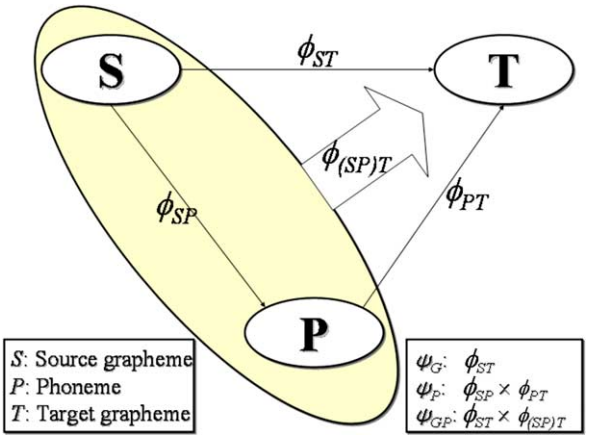

Fig. 1. Three transliteration models.

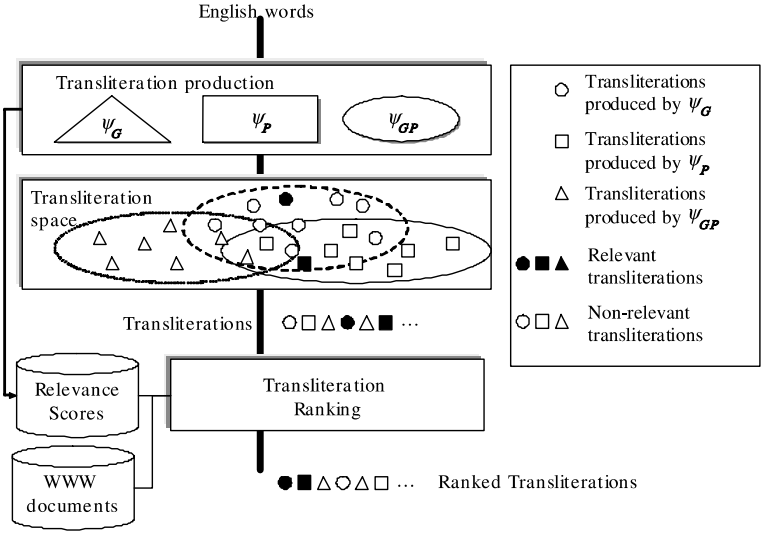

Fig. 2. Basic system architecture of an ensemble-based transliteration model.

and the combination of grapheme-based transliteration and phoneme-based transliteration, 'neomaisin', respectively. The three transliteration models (ψ_G, ψ_P, and ψ_{GP}), therefore, should be considered simultaneously in order to reflect the dynamic behaviors of transliteration.[5]

Fig. 2 shows the basic system architecture of our ensemble-based transliteration model that makes use of three transliteration models to consider the complex process of transliteration. Our model is composed of **transliteration production** and **transliteration ranking**. In the transliteration production step, our model tries to generate transliterations using three different transliteration models. Then the transliterations are ranked with web data and relevance scores given by each transliteration model in the transliteration ranking step. The main goal of the transliteration production step is to produce all possible transliterations reflecting dynamic behaviors of transliteration. In other words, transliteration production tries to make transliterations where relevant transliterations are included as many as possible. The basic assumption of transliteration

[5] In our English-to-Korean transliteration test set (Nam, 1997), we find that about 60% are phoneme-based transliterations, while about 30% are grapheme-based transliterations. The others are transliterations, one part of which is a grapheme-based transliteration and the other of which is a phoneme-based transliteration.

ranking with web data is that relevant transliterations will be more frequently used in real-world texts than non-relevant ones. Web data reflecting the real-world usage of transliterations is used as the language resource to rank the transliterations. If a transliteration appears in more WWW documents, then it gets a higher rank or it gets a higher score. The assumption of transliteration ranking with relevance scores is that the relevant transliteration will get a high relevance score regardless of transliteration models. The goal of transliteration ranking, therefore, is to locate relevant transliterations in the top ranks and to locate non-relevant ones in the bottom ranks.

The rest of this paper is organized as follows: In Sections 2 and 3, we will describe two core parts of our ensemble-based transliteration model. Section 4 shows experiments on machine transliteration and on application of our ensemble-based transliteration model to information retrieval. Section 5 shows discussion about machine transliteration models. We will conclude our paper in Section 6.

2. Transliteration production based on three machine transliteration models

In this paper, we implement three different machine transliteration models (ψ_G, ψ_P, and ψ_{GP}) using several machine learning techniques. Fig. 3 summarizes the differences among the three transliteration models and their component functions. ψ_G is composed of one component function called $\varphi_{ST}: S \rightarrow T$, which directly transforms a source grapheme (S) to a target grapheme (T); while ψ_P is composed of two component functions called $\varphi_{SP}: S \rightarrow P$ and $\varphi_{PT}: P \rightarrow T$ where φ_{SP} is a function for producing pronunciation and φ_{PT} is a function for producing target graphemes based on phonemes (P). Like ψ_P, ψ_{GP} is composed of two functions called $\varphi_{SP}: S \rightarrow P$ and $\varphi_{(SP)T}: S \times P \rightarrow T$. The difference between ψ_P and ψ_{GP} is the function used to produce target graphemes. While ψ_P uses only the phonemes, ψ_{GP} uses source graphemes and its corresponding phonemes when it generates target graphemes. We describe their differences with two functions, $\varphi_{PT}: P \rightarrow T$ and $\varphi_{(SP)T}: S \times P \rightarrow T$.

To train each component function, we should define the features that represent training instance and data. Table 1 shows five feature types, f_S, f_P, f_{GS}, f_{GP}, and f_T. Note that $f_{S,GS}$ is a symbol representing both f_S and

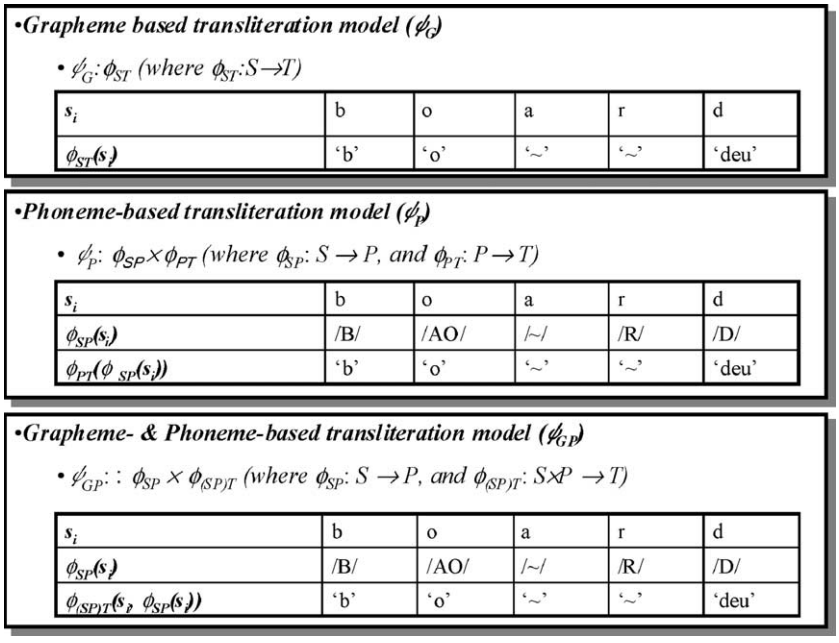

Fig. 3. Three different transliteration models and their component functions: where S is a set of source graphemes (e.g. English alphabets), P is a set of phonemes defined in ARPAbet[a], T is a set of target graphemes. ([a]ARPAbet symbol will be used for representing phonemes. ARPAbet is one of the methods used for coding phonemes into ASCII characters (www.cs.cmu.edu/~laura/pages/arpabet.ps). In this paper, we will denote phonemes and pronunciation with two slashes like so: /AH/. Pronunciation represented in this paper is based on *The CMU Pronunciation Dictionary* and *The American Heritage*(*r*) *Dictionary of the English Language*.)

Table 1
Feature types used for transliteration models

Feature type		Description	Possible feature values
$f_{S,GS}$	f_S	Source graphemes	Source graphemes in S; 26 alphabets for English
	f_{GS}	Source grapheme type	Consonant (C), and Vowel (V)
$f_{P,GP}$	f_P	Phonemes	Phonemes in P (/AA/, /AE/, and so on)
	f_{GP}	Phoneme type	Consonant (C), Vowel (V), Semi-vowel (SV) and silence (/∼/)
	f_T	Target graphemes	Target graphemes in T

Table 2
Definition of each component function

Notation	Used feature types	Input	Output
$\varphi_{SP}: S \rightarrow P$	$f_{S,GS}$, f_P	Source grapheme: s_i	Phoneme: $\varphi_{SP}(s_i)$
$\varphi_{(SP)T}: S \times P \rightarrow T$	$f_{S,GS}$,$f_{P,GP}$, f_T	Correspondence between source grapheme and phoneme: $\langle s_i, p_i = \varphi_{SP}(s_i)\rangle$	Target grapheme: $t_i = \varphi_{(SP)T}(s_i, p_i)$
$\varphi_{PT}: P \rightarrow T$	$f_{P,GP}$, f_T	Phoneme: $p_i = \varphi_{SP}(s_i)$	Target grapheme: $t_i = \varphi_{PT}(p_i)$
$\varphi_{ST}: S \rightarrow T$	$f_{S,GS}$, f_T	Source grapheme: s_i	Target grapheme: $t_i = \varphi_{ST}(s_i)$

f_{GS}, and $f_{S,GS}$ is a symbol representing both f_P and f_{GP}. According to component functions, we use different feature types. How to model each component function with the feature types will be given in Sections 2.1 and 2.2.

2.1. Component functions of each transliteration model

Table 2 shows the definitions of four component functions that we used. They are defined in terms of their input and output. Their role in each transliteration model is to select the most relevant output among output candidates derived from their input. The performance of transliteration models, therefore, strongly depends on their component functions. In other words, how well each component function is modeled is the key to making a good machine transliteration system. The modeling highly depends on the feature type and context.

From the viewpoint of the feature type, φ_{ST}, φ_{PT}, and $\varphi_{(SP)T}$ make use of different feature types as described in Table 2. Depending on the difference, three component functions show their strong points and weak points in machine transliteration. φ_{ST} is good at producing grapheme-based transliterations while it shows weak points for phoneme-based transliterations. On the contrary, φ_{PT} has advantage in producing phoneme-based transliterations but it has disadvantage in producing grapheme-based transliterations. For ''amylase'' and its standard Korean transliteration 'amillaaje', which is a grapheme-based transliteration, φ_{ST} can produce the correct transliteration; while φ_{PT} tends to produce wrong ones like 'aemeolleiseu' derived from /AE M AH L EY S/, which is pronunciation of ''amylase''. On the other hand, φ_{PT} is capable of producing 'deiteo', which is the standard Korean transliteration of ''data'' and a phoneme-based transliteration; while φ_{ST} tends to give wrong ones like 'data'.

$\varphi_{(SP)T}$ adopts merits of φ_{ST}, and φ_{PT} by using correspondence between source grapheme and phoneme. The correspondence gives power for $\varphi_{(SP)T}$ to produce both grapheme-based transliteration and phoneme-based transliteration. Furthermore, the correspondence gives important clues for resolving transliteration ambiguities in many cases.[6] For example, phoneme /AH/ produces high ambiguities in machine transliteration because it can be mapped to almost every single vowels in source language and target language (the underlined grapheme corresponds to /AH/: cinem**a**, host**e**l, hol**o**caust in English, 'sinem**a**', 'host**e**l', 'holl**o**koseuteu' in their Korean counterparts, and 'sinem**a**', 'hoseut**e**ru', 'hor**o**koosuto' in their Japanese counterparts). If we know the correspondence between source grapheme and phoneme in this case, we can more easily infer the correct transliteration of /AH/, because the correct target grapheme corresponding to /AH/ usually depends on the source

[6] Though contextual information can also reduce the ambiguity, we limit our discussion to the feature type in this paragraph.

Table 3
Examples of Korean graphemes derived from source grapheme "a" and its corresponding phoneme: the underline indicates source graphemes corresponding to each phoneme in the phoneme column

Korean grapheme	Phoneme	Example
'a'	/AA/	ad**a**gio, saf**a**ri, viv**a**ce
'ae'	/AE/	adv**a**ntage, **a**lab**a**ster, tr**a**vertine
'ei'	/EY/	ch**a**mber, champ**a**gne, ch**a**os
'i'	/IH/	advant**a**ge, aver**a**ge, sil**a**ge
'o'	/AO/	**a**llspice, b**a**ll, ch**a**lk

grapheme corresponding to /AH/. On the other hand, Korean transliterations of source grapheme "a" are various like 'a', 'ae', 'ei', 'i', and so on. In this case, phonemes corresponding to source grapheme "a" can help a component function to resolve the transliteration ambiguities like Table 3. In Table 3, the underlined source grapheme "a" in the example column is pronounced as the phoneme in the phoneme column. The correct Korean transliterations corresponding to source grapheme "a" can be more easily found, like in the Korean grapheme column, by means of phonemes in the phoneme column.

Though $\varphi_{(SP)T}$ is more effective than both φ_{ST} and φ_{PT} in many cases, $\varphi_{(SP)T}$, sometimes, shows its weak points when the standard transliteration is strongly biased to either grapheme-based transliteration or phoneme-based transliteration. In that case, one of source grapheme and phoneme, which does not contribute on producing the correct transliteration, sometimes, makes it difficult for $\varphi_{(SP)T}$ to effectively produce the correct transliteration using the other.

Because φ_{ST}, φ_{PT}, and $\varphi_{(SP)T}$ are the core part of ψ_G, ψ_P, and ψ_{GP}, respectively, the merits and demerits of the three component functions become those of three transliteration models that each component function participates in. We combine transliterations produced by the three transliteration models in order to make up for the weak points of one transliteration model by the strong points of the others.

Transliteration usually depends on context. For example, source grapheme "a" can be differently transliterated into Korean graphemes according to its context, like 'ei' in the context of "-ation" and 'a' in the context of "art".[7] In making use of context information, determining context window size is important. Too narrow context window can degrade the transliteration performance due to lack of context information. For example, when source grapheme "t" in "-tion" is transliterated into Korean, the right one source grapheme as context is insufficient because the right three contexts of "t", "-ion", are necessary to get the correct target grapheme 'ei'. Too wide context window can also degrade the transliteration performance because it decreases power to resolve transliteration ambiguities. For these reasons, many previous works determine their context window size as 3. In this paper, we also use the same context window size as the previous work's (Goto, Kato, Uratani, & Ehara, 2003; Kang & Choi, 2000). The effect of context window size on transliteration performance will be shown in Section 4 through an experiment.

Table 4 shows how to select the most relevant output in each component function using context information. In the table, L1–L3, R1–R3, and C0 represent the left context, right context, and current context (or focus), respectively. φ_{SP} produces the most relevant phoneme for each source grapheme. Let $SW = s_1 \cdot s_2 \cdots s_n$ be an English word, then SW's pronunciation can be represented as a sequence of phonemes produced by φ_{SP}, such as $P_{SW} = p_1 \cdot p_2 \cdots p_n$ where $p_i = \varphi_{SP}(s_i)$. φ_{SP} is composed of two steps. The first step involves a search in a pronunciation dictionary, which contains English words and their pronunciation. This paper uses *The CMU Pronouncing Dictionary* (CMU, 1997), which contains 120,000 English words and their pronunciation. The second step is to estimate the pronunciation. If an English word is not registered in the pronunciation dictionary, we must estimate its pronunciation. Produced pronunciation is used for φ_{PT} in ψ_P and $\varphi_{(SP)T}$ in ψ_{GP}.

φ_{ST}, φ_{PT}, and $\varphi_{(SP)T}$ produce target graphemes using their input. Like φ_{SP}, the three component functions use their previous outputs represented by f_T. Depending on input and output of each component function

[7] In considering the context window size, the minimal unit of contexts is very important. The minimal unit can be one alphabet like the example, chunk of alphabets like in Kang and Kim (2000) and syllable like in Fujii and Ishikawa (2001) and Goto et al. (2003). Either the chunk of alphabet or the syllable as the minimal unit of contexts makes it possible for a transliteration system to consider larger window size than one alphabet.

Table 4
Framework of each component function: $ represents the start of words

	Feature types	Left context			Focus	Right context				Output
		L3	L2	L1	**C0**	R1	R2	R3		
φ_{SP}	f_S	$	$	$	**b**	o	a	r	→	$\varphi_{SP}(C\underline{0}) =$ /B/
	f_{GS}	$	$	$	**C**	V	V	C		
	f_P	$	$	$						
$\varphi_{(SP)T}$	f_S	$	$	$	**b**	o	a	r	→	$\varphi_{(SP)T}(C\underline{0}) =$ 'b'
	f_P	$	$	$	**/B/**	/AO/	/∼/	/R/		
	f_{GS}	$	$	$	**C**	V	V	C		
	f_{GP}	$	$	$	**C**	V	C	C		
	f_T	$	$	$						
φ_{PT}	f_P	$	$	$	**/B/**	/AO/	/∼/	/R/	→	$\varphi_{PT}(C0) =$ 'b'
	f_{GP}	$	$	$	**C**	V	∼	C		
	f_T	$	$	$						
φ_{ST}	f_S	$	$	$	**b**	o	a	r	→	$\varphi_{ST}(C0) =$ 'b'
	f_{GS}	$	$	$	**C**	V	V	C		
	f_T	$	$	$						

described in Table 2, φ_{ST}, φ_{PT}, and $\varphi_{(SP)T}$ produce a sequence of target graphemes like $T_{SW} = t_1 \cdot t_2 \cdots t_n$ for $SW = s_1 \cdot s_2 \cdots s_n$ and $P_{SW} = p_1 \cdot p_2 \cdots p_n$.

2.2. Machine learning algorithms for each component function

In this section, we will describe a way of modeling component functions using three machine learning algorithms: maximum entropy model, decision tree, and memory-based learning, which are probability based model, rule based model, and vector-based model (or example-based model). Because φ_{ST}, φ_{PT}, and $\varphi_{(SP)T}$ share the similar framework, we limit our focus to $\varphi_{(SP)T}$ among them in this section.

2.2.1. Maximum entropy model

The maximum entropy model (MEM) is a widely used probability model that can incorporate heterogeneous information effectively (Berger, Della Pietra, & Della Pietra, 1996; Miyao & Tsujii, 2002). In the maximum entropy model, an event ev is usually composed of a target event (te) and a history event (he), say $ev = \langle te, he \rangle$. Event ev is represented by a bundle of feature functions, $fe_i(ev)$, which represent the existence of a certain characteristic 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; Miyao & Tsujii, 2002).

φ_{SP} and $\varphi_{(SP)T}$ based on the maximum entropy model can be represented as formula (1). History events in each component function are made from the left, right, and current context. For example, history events for $\varphi_{(SP)T}$ are composed of $f_{S,GS(i-3,i+3)}$, $f_{P,GP(i-3,i+3)}$, and $f_{T(i-3,i-1)}$ where i is the index of the current source grapheme and phoneme to be transliterated and $f_{X(l,m)}$ represents the features of feature type f_X located from the position l to the position m. Target events are a set of outputs derived from history events in each component function. Given history events, $\varphi_{(SP)T}$ (φ_{SP}) finds the most probable target grapheme (phoneme), which maximizes formula (1). One important thing in designing a model based on the maximum entropy model is to determine feature functions which effectively support certain decision of the model. Our basic philosophy of feature function design for each component function is that context information collocated with the unit of interest is an important factor. With the philosophy, we determined the activating conditions (or history events) of the feature functions by combination of features in feature types. Possible feature combinations for history events are between features in the same feature type and between features in the different feature types. The used feature combinations in each component function are listed in Table 5.

In formula (2), history events of φ_{SP} and $\varphi_{(SP)T}$ are represented as he(SP) and he(SPT), respectively. Target events are also represented in the same manner like te(SP) and te(SPT). Because the conditional probability in

Table 5
Used feature combinations for history events

φ_{SP}	$\varphi_{(SP)T}$
• Feature combinations between features in the same feature type • Feature combinations between features in different feature types A. between $f_{S,GS}$ and f_P	• Feature combinations between features in the same feature type • Feature combinations between features in different feature types A. between $f_{S,GS}$ and $f_{P,GP}$ B. between $f_{S,GS}$ and f_T C. between $f_{P,GP}$ and f_T

formula (1) can be represented with feature functions, it can be rewritten as formula (2). Formula (3) shows examples of feature functions for φ_{SP} and $\varphi_{(SP)T}$; Table 4 is used for deriving the feature functions. fe_l represents lth for φ_{SP} and fe_m represents mth feature function for $\varphi_{(SP)T}$, respectively. f_{Si}, f_{Pi}, and f_{Ti} in formula (3) correspond to f_S, f_P, and f_T at C0 in Table 4, respectively. f_{Pi-1} corresponds to f_P at L1 in Table 4. fe_l represents an event where f_{Si} (C0 in φ_{SP}) is "b", f_{Pi-1} (L1 in φ_{SP}) is "$" and f_{Pi} (Output in φ_{SP}) is "/B/". fe_m represents an event where f_{Si} (C0 in $\varphi_{(SP)T}$) is "b", f_{Pi} (C0 in $\varphi_{(SP)T}$) is "/B/" and f_{Ti} (Output in $\varphi_{(SP)T}$). In order to model each component function based on MEM, Zhang's maximum entropy modeling tool is used (Zhang, 2004).

$$\begin{aligned}\varphi_{(SP)T}(s_i,p_i) &= \arg\max p(t_i|f_{T(i-3,i-1)},f_{S,GS(i-3,i+3)},f_{P,GP(i-3,i+3)})\\ \varphi_{SP}(s_i) &= \arg\max p(p_i|f_{P(i-3,i-1)},f_{S,GS(i-3,i+3)})\end{aligned} \tag{1}$$

$$\begin{aligned}p(t_i|f_{T(i-3,i-1)},f_{S,GS(i-3,i+3)},f_{P,GP(i-3,i+3)}) &= \frac{1}{Z_h(SPT)}\prod_j \alpha_j^{fe_j(te(SPT),he(SPT))}\\ p(p_i|f_{P(i-3,i-1)},f_{S,GS(i-3,i+3)}) &= \frac{1}{Z_h(SP)}\prod_k \beta_k^{fe_k(te(SP),he(SP))}\end{aligned} \tag{2}$$

$$\begin{aligned}fe_l(te(SP),he(SP)) &= \begin{cases}1 & \text{if } f_{Si}=b, f_{Pi-1}=\$ \text{ and } f_{Pi}=/B/\\ 0 & \text{otherwise}\end{cases}\\ fe_m(te(SPT),he(SPT)) &= \begin{cases}1 & \text{if } f_{Si}=b, f_{Pi}=/B/ \text{ and } f_{Ti}='b'\\ 0 & \text{otherwise}\end{cases}\end{aligned} \tag{3}$$

2.2.2. Decision tree

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

Training data for each component function is represented by features of feature types in the context of L3–L1, C0, and R1–R3 as described in Table 4. C4.5 tries to construct decision tree by looking for regularities in the training data (Mitchell, 1997). Fig. 4 shows a fraction of constructed decision tree for φ_{SP} and $\varphi_{(SP)T}$ in English-to-Korean transliteration (note that the left side represents the decision tree for φ_{SP} and the right side represents the decision tree for $\varphi_{(SP)T}$). A set of the target classes in the decision tree for φ_{SP} will be a set of phonemes and that for $\varphi_{(SP)T}$ will be a set of target graphemes. In the figure, rectangles indicate a leaf node, which describes the target class, and circles indicate a decision node. In order to simplify our examples, we use only the f_S and f_P. (Note that actually all feature types for each component function as described in Table 4 are used for constructing decision trees.) Intuitively, the most effective feature for φ_{SP} and $\varphi_{(SP)T}$ may be located in C0 among L3–L1, C0, and R1–R3 because the correct outputs of φ_{SP} and $\varphi_{(SP)T}$ strongly depend on source grapheme or phoneme in the C0 position. As we expected, the most effective feature in the decision trees is located in the C0 position like C0(f_S) for φ_{SP} and C0(f_P) for $\varphi_{(SP)T}$. (Note that the first feature to be tested is the most effective feature in decision trees.) In Fig. 4, the decision tree for φ_{SP} outputs phoneme /AO/ for the instance x(SP) by retrieving the decision nodes with a sequence of C(f_S) = o, R1(f_S) = a, and R2(f_S) = r represented with '*'. With the sim-

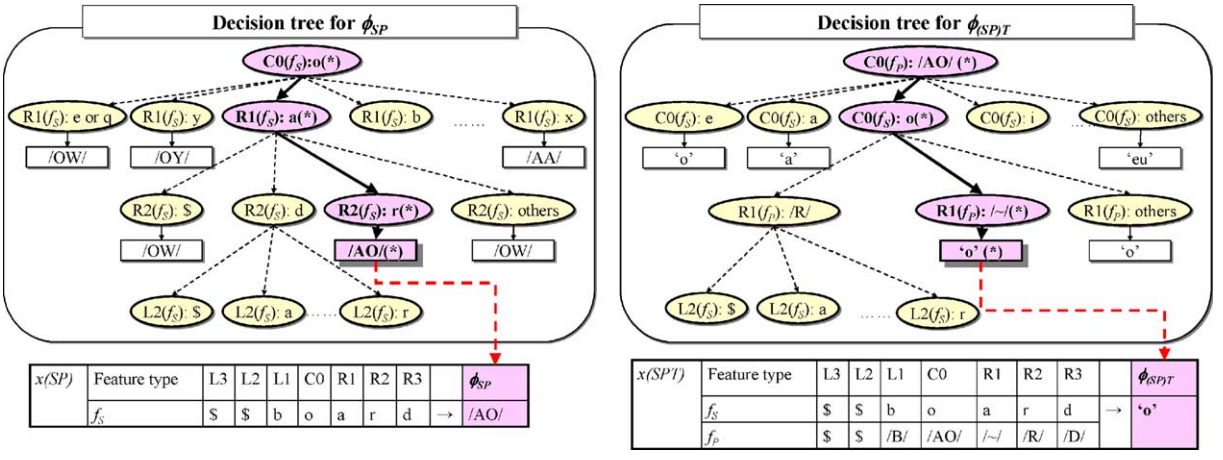

Fig. 4. Decision trees for φ_{SP} and $\varphi_{(SP)T}$.

ilar manner, the decision tree for $\varphi_{(SP)T}$ produces target grapheme (Korean grapheme) 'o' for the instance x(SPT) by retrieving the decision nodes from C0(f_P) = /AO/ to R1(f_P) = /~/ represented with '*'.

2.2.3. Memory-based learning

Memory-based learning (MBL) is an example-based learning method. It is also called instance-based learning and case-based learning method. It is based on a k-nearest neighborhood algorithm (Aha, 1997; Aha, Kibler, & Albert, 1991; Cover & Hart, 1967; Devijver & Kittler, 1982). MBL represents a training data as a vector. In the training phase, MBL puts all training data as examples in a memory, and clusters some examples with a k-nearest neighborhood principle. It then produces an output using similarity-based reasoning between test data and examples in the memory. Let test data be x and a set of examples in a memory be Y, the similarity between x and Y is estimated by a distance function, $\Delta(x, Y)$. MBL selects an example y_i or a cluster of examples that are most similar to x, then assigns a target class of the example to x's target class. We use a memory-based learning tool called TiMBL (Tilburg memory-based learner) version 5.0 (Daelemans et al., 2002).

Training data for MBL is represented in the same form as training data for decision tree. Note that the target classes for φ_{SP} and $\varphi_{(SP)T}$, which MBL outputs, are phoneme and target grapheme, respectively. Like Fig. 4, we use only the f_S and f_P in order to simplify our examples. Fig. 5 shows examples of φ_{SP} and $\varphi_{(SP)T}$ based on MBL for English-to-Korean transliteration. All training data are represented with their features in the context of L3–L1, C0, and R1–R3 and their target classes for φ_{SP} and $\varphi_{(SP)T}$. They are stored in the memory through a training phase. Feature weighting for dealing with features of differing importance is also per-

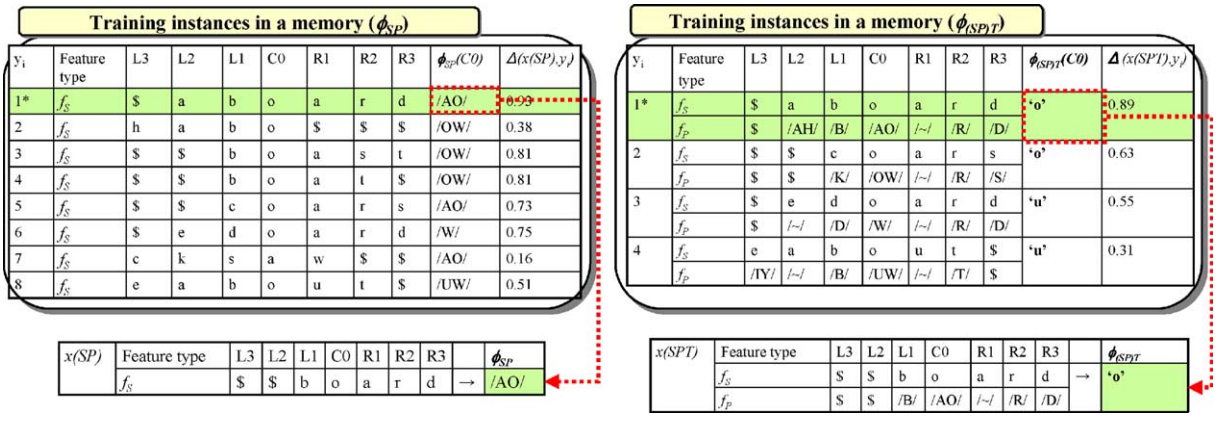

Fig. 5. Memory-based learning for φ_{SP} and $\varphi_{(SP)T}$.

formed in the training phase. In Fig. 5, φ_{SP} based on MBL outputs the phoneme /AO/ for x(SP) by comparing the similarities between x(SP) and Y using distance metric $\Delta(x(SP), Y)$. With the similar manner, $\varphi_{(SP)T}$ based on MBL outputs the target grapheme 'o'.

3. Transliteration ranking

In this paper, we use two kinds of ranking method, relevance-based transliteration ranking and web-based transliteration ranking. Once transliterations are produced for a given English word through three different transliteration models, the transliterations are ranked by the two ranking methods.

3.1. Web-based transliteration ranking

The assumption of transliteration ranking with web data is that relevant transliterations will more frequently appear in WWW documents than non-relevant ones do. Grefenstette (1999) used phrase web frequency to disambiguate all possible English translations of German and Spanish compound nouns. With the similar manner, we use web frequency (the number of WWW documents where transliterations exist) for transliteration ranking.

It is important to consider a transliteration and its source language word at a time rather than a transliteration alone in order to get reliable web frequencies, because our aim in the transliteration ranking is to find relevant transliterations corresponding to source language word rather than to find transliterations, which are frequently used in the target language. Therefore, the most desirable case is to find WWW documents where transliterations are used as translations of the source language word. **Bilingual Phrasal Search** (BPS), where a phrase composed of a transliteration and its source language word is used as a query of a search engine such as {'amillaaje' "amylase"}, enables a search engine to find the WWW documents that fit our aim. Fig. 6 shows Korean and Japanese WWW documents retrieved by the BPS for "amylase" and its Korean/Japanese transliterations, 'amillaaje' and 'amiraaje'. The retrieved WWW documents usually contain a transliteration and its source language word as a translation pair in the parenthesis expressions, like the rectangles in Fig. 6.

Fig. 6. Desirable retrieved web pages for transliteration ranking.

There is dilemma between the quality and coverage of retrieved WWW documents. Though BPS gives high-quality WWW documents that fit our aim of transliteration ranking, the coverage of BPS, however, is relatively low. It means that we cannot retrieve WWW documents for some transliterations with BPS. For example, BPS for <"alkalosis" 'arukarosisu'> and <"ermine" 'eomin'> retrieved no WWW document. Therefore, the alternative search methods are necessary when BPS fails to retrieve WWW documents. **Bilingual Keyword Search** (BKS), which is first applied when BPS fails, and **Monolingual Keyword Search** (MKS), which is used when both BPS and BKS fail, are the alternative search methods.

Like BPS, BKS makes use of two keywords including a transliteration and its source language word as a query of a search engine. BPS retrieves WWW documents where two keywords exist as a phrase, but BKS retrieves WWW documents if the two keywords exist just in the same document. Due to this property of BKS, web frequencies of noisy transliterations are sometimes higher than those of relevant transliterations, especially when the noisy transliterations are one-syllable transliterations. For example, 'mok', which is one of produced Korean transliterations of "mook" and is the one-syllable noisy transliteration, gets higher web frequency than 'mukeu', which is the standard transliteration of "mook", because 'mok' is a common noun, which frequently appears in Korean texts as the meaning of "neck". However, BKS can improve coverage without great loss of quality of retrieved WWW documents if transliterations are composed of more than two syllables.

Though BKS has higher coverage than BPS does, BKS also fails to retrieve WWW documents for some cases. For this case, MKS is applied. In MKS, a transliteration alone is used as a query for a search engine. Compared with translation model and language model in machine translation, both BPS and BKS can be regarded as the translation model; while MKS plays a role as the language model. Though MKS does not give information whether a transliteration is the correct transliteration corresponding to a given source language word, it can give information whether a transliteration tends to be a target language word. The three search methods are sequentially applied from BPS to MKS until one of them retrieves at least one WWW documents for transliterations corresponding to one source language word; If one search method fails to retrieve WWW documents for all transliterations, then the next search method is applied.

Along with three different search strategies described above, three different search engines are used in order to get more WWW documents. Search engines to be used should satisfy two conditions: (1) to support Korean/Japanese WWW document retrieval, and (2) to support phrasal search as well as keyword search. Google,[8] Yahoo,[9] and MSN[10] that satisfy the two conditions are chosen as search engines.

Web frequencies acquired from the three search methods and the three search engines are used as formula (4) where c_i is the ith transliteration produced by three transliteration models, e is c_i's source language word, RF is a ranking function for transliterations, WF is a function for web frequency, NWF is a function for normalized web frequency, C is a set of produced transliterations and j is an index for the jth search engine. We use normalized web frequency (NWF) as a ranking factor (RF). Normalized web frequency (NWF) is web frequency (WF) divided by the total web frequency of all transliterations corresponding to one source language word. Then the score of each transliteration is calculated by summing up the normalized web frequency of the transliteration given by the three search engines:

$$S_{\mathrm{Web}}(e,c_i)=\begin{cases}\mathrm{RF}_{\mathrm{BPS}}(e,c_i)=\sum_j \mathrm{NWF}_{\mathrm{BPS}_j}(e,c_i) & \text{if } \sum_j \sum_{c_k\in C} \mathrm{WF}_{\mathrm{BPS}_j}(e,c_k)\neq 0\\ \mathrm{RF}_{\mathrm{BKS}}(e,c_i)=\sum_j \mathrm{NWF}_{\mathrm{BKS}_j}(e,c_i) & \text{if } \sum_j \sum_{c_k\in C} \mathrm{WF}_{\mathrm{BPS}_j}(e,c_k)=0\\ \mathrm{RF}_{\mathrm{MKS}}(e,c_i)=\sum_j \mathrm{NWF}_{\mathrm{MKS}_j}(e,c_i) & \text{if } \sum_j \sum_{c_k\in C} \mathrm{WF}_{\mathrm{BPS}_j}(e,c_k)=\sum_j \sum_{c_k\in C} \mathrm{WF}_{\mathrm{BKS}_j}(e,c_k)=0\end{cases} \tag{4}$$

$$\text{where } \mathrm{NWF}_j(e,c_i)=\frac{\mathrm{WF}_j(e,c_i)}{\sum_{c_k\in C}\mathrm{WF}_j(e,c_k)}$$

Table 6 shows the ranking example for English word "data" and its possible Korean transliterations, 'deiteo', 'deita', and 'deta'. In the table, the normalized $\mathrm{WF}_{\mathrm{BPS}}$ ($\mathrm{NWF}_{\mathrm{BPS}}$) for search engine **A** is calculated as follows:

[8] http://www.google.com.
[9] http://www.yahoo.com.
[10] http://www.msn.com.

Table 6
An example of ranking for "data"

Search engine	e = "data"						Total frequency
	c_1 = 'deiteo'		c_2 = 'deita'		c_3 = 'deta'		
	WF_{BPS}	NWF_{BPS}	WF_{BPS}	NWF_{BPS}	WF_{BPS}	NWF_{BPS}	
A	94,100	0.5811	67,800	0.4186	54	0.0003	161,954
B	101,834	0.7957	26,132	0.2042	11	0.0001	127,977
C	1358	0.3080	3028	0.6868	23	0.0052	4409
RF_{BPS}		1.6848		1.3096		0.0056	

- NWF_{BPS}("data", 'deiteo') = 94100/(94100 + 67800 + 54) = 0.5811.
- NWF_{BPS}("data", 'deita') = 67800/(94100 + 67800 + 54) = 0.4186.
- NWF_{BPS}("data", 'deta') = 54/(94100 + 67800 + 54) = 0.0003.

Then, the ranking score of 'deiteo' is calculated by summing up NWF_{BPS}("data", 'deiteo') of each search engine as follows:

- RF_{BPS}("data", 'deiteo') = 0.5810 + 0.7957 + 0.3080 = 1.6848.

Using the RF value for each transliteration, we can make ranked transliterations: c_1 is the first place, c_2 is the second place, and c_3 is the last place.

3.2. Relevance-based transliteration ranking

Though the web-based transliteration ranking method may be effective, it shows limitations when there is no web data containing transliterations to be ranked. Therefore we need another ranking method which can rank transliterations independent of web-data in order to handle such the case. Our relevance-based transliteration ranking method makes use of a relevance score given by each transliteration model. When each transliteration model produces transliterations for source words, it gives their relevance score from the viewpoint of the transliteration model. The three transliteration models give their own relevance scores to produced transliterations and the scores are used as a ranking factor in relevance-based transliteration ranking. Let e be a source word, c_i be the ith transliteration derived from e, $SR(\psi, e, c_i)$ be the relevance score of c_i given by the transliteration model ψ and $S_{Rel}(e, c_i)$ be the relevance score of c_i. Then $S_{Rel}(e, c_i)$ can be defined as formula (5). The assumption of relevance-based ranking method is that the relevant transliteration will get a high relevance score regardless of transliteration model. Therefore we get $S_{Rel}(e, c_i)$ by summing up $SR(\psi, e, c_i)$. Note that the relevance score of c_i for each transliteration model ($SR(\psi, e, c_i)$ is normalized with $\sum_i SR(\psi, e, c_i)$.

$$S_{Rel}(e, c_i) = \sum_{\psi \in \{\psi_G, \psi_P, \psi_{GP}\}} \frac{SR(\psi, e, c_i)}{\sum_i SR(\psi, e, c_i)} \tag{5}$$

Finally, transliterations are ranked by both the web-based transliteration ranking method and the relevance-based transliteration ranking method like formula (6), where λ ($0 \leqslant \lambda \leqslant 1$) is a weighting parameter for $S_{Web}(e, c_i)$ and $S_{Rel}(e, c_i)$.

$$S(e, c_i) = \lambda \times S_{Web}(e, c_i) + (1 - \lambda) \times S_{Rel}(e, c_i) \tag{6}$$

4. Experiments and evaluation

4.1. Experiments on machine transliteration

4.1.1. Experimental setup

We perform experiments for English-to-Korean and English-to-Japanese transliteration. English-to-Korean test set (EKSet) (Nam, 1997) consists of 7185 English–Korean pairs—the number of training data is

6185 and that of test data is 1000. EKSet contains no transliteration variation; therefore, there is one transliteration for an English word. English-to-Japanese test set (EJSet), which is an English–Katakana pair in EDICT[11] (Breen, 2003), consists of 10,398 pairs—1000 for test and the rest for training. EJSet contains transliteration variations, like <micro, 'maikuro'>, and <micro, 'mikuro'>; the average number of Japanese transliterations for an English word is 1.15. Evaluation is performed by word accuracy (W.A.), which was used as the evaluation measure in the previous works:

$$\text{W.A.} = \frac{\text{the number of correct words}}{\text{the number of generated words}} \tag{7}$$

We perform four experiments as follows:

- ***Comparison Test I***: Performance evaluation of grapheme- and phoneme-based transliteration model and the previous works (to investigate the effect of correspondence between source grapheme and phoneme on machine transliteration).
- ***Comparison Test II***: Performance evaluation of **transliteration production** for each transliteration model (to investigate how well our system produces transliterations and to investigate contribution of each machine transliteration model to **transliteration production**).
- ***Context Window Size Test***: Performance evaluation depending on context window size (to investigate the effect of context window size on the performance of machine transliteration).
- ***Ranking Test***: Performance evaluation of **transliteration ranking** (to investigate how well our system ranks transliterations).

4.1.2. Experimental results

Table 7 shows results of **Comparison Test I**. In the table, MEM, DT, and MBL represent maximum entropy model, decision tree, and memory-based learning, respectively. GDT (Kang, 2001; Kang & Choi, 2000), GPC (Kang & Kim, 2000), GMEM (Goto et al., 2003), and HWFST (Bilac & Tanaka, 2004), which are one of machine transliteration methods showing good performance in English-to-Korean transliteration and English-to-Japanese transliteration, are compared with ψ_{GP}. They are trained and tested with the same data as ψ_{GP}. In GDT (the decision tree based transliteration method), decision trees, which transform each source grapheme to target graphemes, are learned and then they are straightly applied to machine transliteration. GPC and GMEM are the transliteration network based method for English-to-Korean and English-to-Japanese transliteration, respectively. They share the similar framework in constructing a transliteration network composed of nodes and arcs. A node represents a chunk of source graphemes and its corresponding target grapheme. An arc represents a possible link between nodes and it has a weight showing its strength. HWFST is the hybrid model-based transliteration methods. In the HWFST, ψ_G and ψ_P are modeled with WFSTs and then ψ_G and ψ_P are combined through the linear interpolation style. Imp. means the performance improvement of each transliteration method compared to GDT. Totally, ψ_{GP} shows significant performance improvement, about 15–23%, in English-to-Korean transliteration, and about 15–43% in English-to-Japanese transliteration. Comparison between previous works and ψ_{GP} will be deeply discussed in Section 5. Among MBL, MEM, and DT in ψ_{GP}, MBL shows the best performance because it shows better abilities to handle transliterations of source words containing idiomatic sequence of alphabets such as "-tion", "-ment", "-ware", and so on than the others. The ability may be deceived from the fact that MBL is example-based learning method and vector-based model.

Table 8 shows results of **Comparison Test II** for EKSet and EJSet. The table summarizes the performance of each transliteration model based on three different machine-learning algorithms. The performance of **transliteration production** is represented as $\psi_G + \psi_P + \psi_{GP}$, which indicates the performance when transliterations produced by the three transliteration models are combined.

[11] http://www.csse.monash.edu.au/~jwb/j_edict.html.

Table 7
Results of Comparison Test I

	Method	EKSet		EJSet	
		W.A. (%)	Imp. (%)	W.A. (%)	Imp. (%)
Previous works	GDT	51.4	–	50.3	–
	GPC	55.1	7.20	53.2	5.77
	GMEM	55.9	8.75	56.2	11.73
	HWFST	58.3	13.42	62.5	24.25
ψ_{GP}	MEM	63.3	23.15	67.0	33.20
	DT	62.0	20.62	66.8	32.80
	MBL	66.9	30.16	72.2	43.54

Table 8
Results of Comparison Test II

	EKSet			EJSet		
	MEM (%)	DT (%)	MBL (%)	MEM (%)	DT (%)	MBL (%)
ψ_G	57.1	55.2	55.7	57.0	58.1	63.0
ψ_P	54.2	52.0	54.7	58.0	57.8	62.5
ψ_{GP}	63.3	62.0	66.9	67.0	66.8	72.2
$\psi_G + \psi_P + \psi_{GP}$	73.0	73.2	78.9	77.2	76.4	80.0

In comparison among ψ_G, ψ_P, and ψ_{GP}, ψ_G and ψ_P depend on either source grapheme or phoneme, while ψ_{GP} dynamically use both source grapheme and phoneme. In other words, ψ_G and ψ_P may make an error when either source grapheme or phoneme that they did not consider holds the key for producing a correct transliteration result. For this reason, ψ_{GP} gives the best result among the three transliteration models. From the viewpoint of **transliteration production**, we find that ψ_G, ψ_P, and ψ_{GP} are complementary transliteration models for correct transliteration production. The correct transliteration production means that at least one is correct among all of produced transliterations. High performance in transliteration production can be acquired if a correct transliteration, which one model fails to produce, can be acquired by the others. By combining three transliteration models, we additionally improve 15–43% W.A. in English-to-Korean and 10–35% in English-to-Japanese compared with ψ_G, ψ_P, and ψ_{GP} used alone in transliteration production.

Table 9 shows the results of **Context Window Size Test**. We use MBL based method, which shows the best performance among machine learning algorithms. Experiments are performed by changing the size of context window from 1 to 6. The result indicates that the best performance is shown when the context window size is 3. When the context window size is 1, there are many cases where the correct transliterations are not produced due to lack of contextual information. For example, in order to produce correct target language grapheme of "t" in "-tion" we need the right three grapheme of "t", "-ion". When the context window size is over 3, it becomes difficult to look for regularities in the training data. Therefore, our system shows the best result when the context window size is 3.

Table 9
Results of Context Window Size Test

Context size	EKSet			EJSet		
	ψ_G (%)	ψ_P (%)	ψ_{GP} (%)	ψ_G (%)	ψ_P (%)	ψ_{GP} (%)
1	46.6	43.9	54.5	48.9	52.8	62.7
2	54.0	53.2	63.3	62.4	60.3	70.0
3	55.7	54.7	66.9	63.0	62.5	72.2
4	54.1	53.7	63.9	62.2	60.5	70.7
5	53.6	53.3	63.8	61.8	58.9	69.3
6	53.9	54.2	63.9	60.0	58.5	69.8

Table 10
Results of Ranking Test

		$S_{Rel}(e, c_i)$		$S_{Web}(e, c_i)$		$S(e, c_i)$	
		EKSet (%)	EJSet (%)	EKSet (%)	EJSet (%)	EKSet (%)	EJSet (%)
ALL	*Top-1*	64.5	64.5	77.1	78.9	77.1	78.9
	Top-2	75.3	76.2	78.9	79.7	78.9	79.7
	Top-3	77.6	79.2	78.9	80.0	78.9	80.0
CTC	*Top-1*	81.75	80.63	97.97	98.6	97.97	98.6
	Top-2	95.44	95.25	100	99.6	100	99.6
	Top-3	98.35	99.00	100	100	100	100

Table 10 shows results of **Rank Test**. In this test, we use MBL, which shows the best performance among the three machine-learning algorithms. We evaluate the performance of the three transliteration ranking methods ($S_{Rel}(e, c_i)$, $S_{Web}(e, c_i)$, $S(e, c_i)$) in two conditions: (1) with all test data (ALL) and (2) with test data for which the transliteration production step makes no errors (CTC). In this test, $\lambda = 0.8$ is used for $S(e, c_i)$. Evaluation with ALL will show the overall performance of our ensemble-based machine transliteration model. Evaluation with CTC will show the performance of the transliteration ranking step when transliteration production makes no error. In the table, the Top-*n* considers whether the correct transliteration is in the Top-*n* ranked transliterations. The average number of produced Korean transliterations is 4.43 and that of Japanese ones is 4.12; note that ψ_P and ψ_{GP} produce more than one transliteration because of pronunciation variations. The ranked results in both English-to-Korean transliteration and English-to-Japanese transliteration indicate that our transliteration ranking method effectively filters out noisy transliterations and locates the correct transliterations in the top rank; most of the correct transliterations are located in Top-1. The overall performance of our machine transliteration system (for ALL) as well as the performance of transliteration ranking (for CTC) is relatively good. Especially, the performance of CTC shows that web data as language resources to rank transliterations is very useful. In comparison between S_{Rel} and S_{Web}, S_{Web} shows more powerful ability in transliteration ranking. However, S_{Rel} makes it possible for a transliteration system to rank transliterations even if S_{Web} can not assign rank score to a transliteration because of absence of web data containing the transliteration to be ranked.

4.1.3. Analysis of results (should be revised depending on three ranking methods)

We define two error types called **production error** and **ranking error**. The production error happens when there is no correct one among all of produced transliterations. The ranking error happens when the standard transliteration does not appear in Top-1 of ranked transliterations.

We examined the effects of BPS, BKS, and MKS on transliteration ranking. Table 11 shows the performance of our method when transliterations are ranked by each search method. In Table 11, **RTC** represents the number of test data which does not cause production error and **NTC** represents the number of test data ranked by each search method. NTC will show the coverage of each search method and the proportion of RTC to NTC will show the upper bound of the performance when each search method is applied. Difference between RTC and NTC will show the number of production errors.

Table 11
Performance according to search methods

	EKSet			EJSet		
	BPS	BKS	MKS	BPS	BKS	MKS
Top-1	86.72%	28.00%	11.93%	90.35%	34.375%	4.11%
Top-2	88.45%	28.00%	14.68%	90.73%	34.375%	5.48%
Top-3	88.45%	28.00%	14.68%	90.73%	35.94%	5.48%
RTC	766	7	16	783	23	4
NTC	866	25	109	863	64	73

Table 12
Some examples cause ranking errors when BPS is applied

English word	Standard transliteration	Web Frequency	Transliteration variations	Web frequency
plaid	'peullaedeu'	93 (by BKS)	'peulleideu'	517 (by BKS)
rosemary	'lojeumeli'	84 (by BPS)	'lojeumali'	504 (by BPS)
indigo	'injigo'	4 (by BPS)	'indigo'	1960 (by BPS)
installation	'insutoleeshon'	9 (by BPS)	'insutaleeshon'	12 (by BPS)

In case of BPS, the highest performance is recorded among the three search methods. BPS handles about 860 cases among 1000 cases that is the total number of test data in EKSet and EJSet. BPS tends to retrieve desirable WWW documents containing transliteration pairs in parenthesis expressions as described in Fig. 6. This property leads us to analyze the ranking errors. We find that the main reason of the ranking errors in BPS is transliteration variations. The transliteration variations contribute to the ranking errors in two aspects. First, when web frequencies of transliteration variations are higher than those of the standard ones, the rank of the variations will be higher than that of the standard one. Table 12 shows some examples of ranking errors caused by this reason. Second, when only the transliteration variations exist but the correct or standard transliterations are not in produced transliterations, ranking errors occurs due to absence of the standard transliteration in produced transliterations. In this case, ranking errors are caused by production errors. Eighty-six cases in EKSet and 71 cases in EJSet are caused by this reason.

In case of BKS and MKS, the number of NTC is relatively small. Because BPS retrieves WWW documents if possible, BKS and MKS can deal with only small number of cases. Table 11 shows that the main reason why BPS fails to retrieve WWW documents is the production error (Note that most cases dealt with BKS and MKS show production error except 23 (7 + 16)[12] cases in EKSet and 27 (23 + 4)[13] cases in EJSet). The result also shows that BKS is more effective than MKS.

Tradeoff between the quality and coverage of retrieved WWW documents is an important factor in transliteration ranking. BPS preferring the quality to the coverage of retrieved WWW documents effectively plays its role with high quality and reasonable coverage.

4.2. Experiments on information retrieval using machine transliteration results

4.2.1. Experimental setup

We investigate the effects of our machine transliteration system on information retrieval. KT SET 2.0 (Park, Choi, Kim, & Kim, 1996), Korean information retrieval test set, and NTCIR-1 test collection[14] (Kando et al., 1999), Japanese information retrieval test set, are used in this experiment. Table 13 shows some statistics of the two test collections. The KTSET contains 2725 kinds of Korean transliterations corresponding to 2391 source language words in documents. The queries contain simple keywords. NTCIR-1 test collection contains about 130,000 kinds of Japanese transliterations corresponding to about 110,000 source language words in documents. The queries are composed of four parts including title, description, narrative and concept, which represent title of a query, short description of a query, long description of a query, and concept keyword of a query, respectively. Among four types of query, the concept query is chosen for Japanese information retrieval in our experiment because it partly contains bilingual information and it is the most effective one among short queries (title, description, and concept).

Because the test collections are for monolingual information retrieval, it is difficult to directly apply our transliteration system to information retrieval experiment. Therefore, we recover the local language query (**L-Query**), which is the original query in the test collections, by substituting transliterations back into their corresponding English words manually, (back-transliterated query: **B-Query**) and then automatically transliterates the English words in B-Query into Korean or Japanese using our machine transliteration system (trans-

[12] 7 (RTC of BKS in EKSet) + 16 (RTC of MKS in EKSet) = 23.
[13] 23 (RTC of BKS in EJSet) + 4 (RTC of MKS in EJSet) = 27.
[14] http://research.nii.ac.jp/ntcir/index-en.html.

Table 13
The statistics of KT SET 2.0 and NTCIR-1 test collections

	KT SET	NTCIR-1
The number of documents	4414	339501
The number of queries	50	83
The average number of relevant documents for each query	28.44	60.36
The average length of documents	150	338.30
The average length of query	2.84	19.69 (concept)

literated local language query: **T-Query**). Table 14 shows examples of L-Query, B-Query, and T-Query for Korean and Japanese test collections. Note that pure Korean and Japanese nouns in the query are not substituted into English words during constructing B-Query from L-Query, such as "網形態" in Japanese. Transliterations in the T-Query should appear in at least one WWW document; in other words, their ranking score given by S_{Web} in transliteration ranking is above 0. Note that T-Query does not always include transliterations (or query terms) in L-Query, because sometimes a transliteration system can not produce the same transliterations in L-Query from English words in B-Query. However, in our experiment, T-Query always includes all query terms in L-Query because our transliteration system always produces the same transliteration in L-Query. Sometimes our system produces transliteration variations, which does not exist in L-Query. Therefore, T-query can be regarded as expanded queries of L-Query.

In this experiment, the performance of information retrieval system when L-Query, B-Query, and T-Query are applied is compared. The purpose of the comparison between L-Query and T-query is to show the effect of transliteration variations on information retrieval system. Because T-Query and L-Query share the same query terms except transliteration variation, which only the T-Query contains, T-Query will show the performance of information retrieval when transliteration variations, which our transliteration system produces, are considered. The purpose of the comparison between B-Query and T-query is to show the effect of machine transliteration on information retrieval system. B-Query and T-Query share the pure Korean query terms or pure Japanese query terms and B-Query contains English query terms corresponding to transliterations or transliteration variations in T-Query. Therefore, the difference between the B-Query and T-Query makes it possible to investigate the performance change of information retrieval system when our machine transliteration system is applied to English query terms.

For information retrieval, Lemur system version 3.1 is used (Paul & Callan, 2001). The retrieved documents are evaluated with interpolated 11 point average precision (Salton, 1989).

4.2.2. Experimental results

Table 15 shows the performance of Korean information retrieval and Japanese information retrieval when L-Query, B-Query, and T-Query are used. Improved % means the performance improvement compared to L-Query. Transliteration variations in T-Query contribute to 10.6% improvement for Korean and 7.6% improve-

Table 14
The examples of Korean and Japanese Queries: underlined words represent transliterations

	L-Query	B-Query	T-Query
Korean	RISC 'weokeuseuteisyeon'	RISC workstation	RISC 'liseukeu' 'weokeuseuteisyeon' 'weokseuteisyeon'
Japanese	'topologii'	Topology	'topologii' 'topologi'
	網形態 'nettowaaku'	網形態 Network	網形態 'nettowaaku' 'nettoowaaku'
	'intaanetto'	Internet	'intaanetto' 'intanetto'

Table 15
The performance of Korean and Japanese information retrieval

		11 pt avg. precision	Improved %
Korean	L-Query	0.3572	–
	B-Query	0.3036	−15.00
	T-Query	0.3950	10.56
Japanese	L-Query	0.3272	–
	B-Query	0.2218	−32.21
	T-Query	0.3521	7.62

ment for Japanese. Our machine transliteration system (comparison between B-Query and T-Query) contributes 30.11% performance improvement for Korean and 58.75% improvement for Japanese. The results indicate that our transliteration system contribute significant performance improvement in information retrieval system by automatically producing transliterations and transliteration variations.

Fig. 7 shows the performance changes depending on the retrieval models in Lemur system. We examine the results retrieved with the following six retrieval models: (1) TFIDF retrieval model (TFIDF), (2) Okapi BM25 retrieval function (OKAPI), (3) KL-divergence language model based retrieval method (KL), (4) InQuery retrieval model (INQUERY), (5) CORI collection selection (CORI_CS), (6) Cosine similarity model (COS). Results show that OKAPI gives the best performance in both languages. The performance of T-Query is higher than that of L-Query and B-Query regardless of retrieval models and languages. Transliterations and transliteration variations produced by our ensemble-based transliteration model, therefore, is effective for performance improvement of information retrieval system.

4.2.3. Analysis of results

We examined the results depending on difference between L-Query and T-Query. If query terms in L-Query and T-Query are different, then they are LTD otherwise LTS. There are two conditions where L-Query and T-Query are LTS. First, there is no English term in B-Query. This means that L-Query and T-Query are composed of only the pure Korean or Japanese terms. Second, query terms, which are transliterations, in L-Query and T-Query are the same. This means that our transliteration system does not produce transliteration variations, which do not exist in L-Query. In our experiment, T-Query always includes all query terms in L-Query when T-Query and L-Query are LTD. Because only the LTD can affect the performance change, we will focus on results of LTD queries rather than LTS queries. The number of LTD is 28 for Korean and 51 for Japanese and that of LTS is 22 for Korean and 32 for Japanese, in our experiment.

For example, the Korean L-Query (“RISC”, ‘weokeuseuteisyeon’) and T-Query (“RISC”, ‘liseukeu’ ‘weokeuseuteisyeon’, ‘weokseuteisyeon’) is an LTD query. In the T-Query, the transliteration of “RISC” (‘liseukeu’) and the transliteration variation of ‘weokeuseuteisyeon’ (‘weokseuteisyeon’) are the query terms that differentiate the L-Query and the T-Query. The L-Query retrieves 16 relevant documents while the T-Query does 24 relevant documents, among 26 relevant documents in the test collection for this query. 11 pt avg. pre-

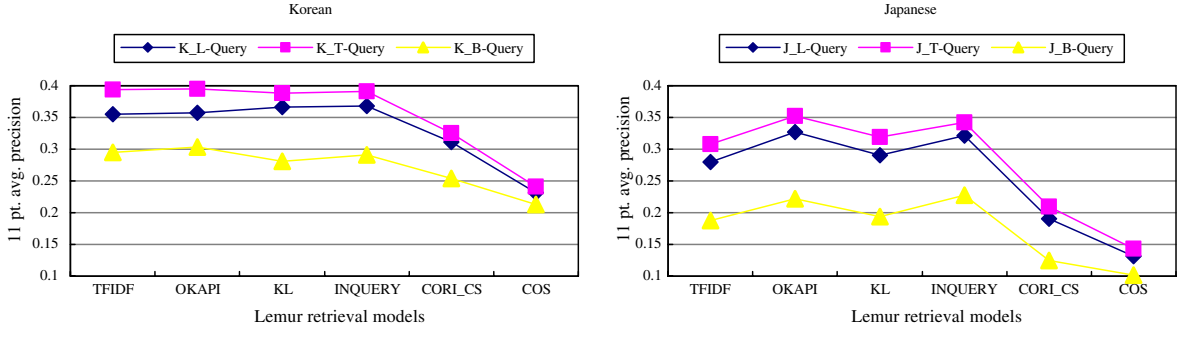

Fig. 7. The performance changes depending on the six retrieval models in the Lemur system.

Table 16
TF and DF of T-Query terms

Query terms	TF (Rel)	DF (Rel)	TF (All)	DF (All)
RISC	59	20	117	64
'liseukeu'	46	16	128	50
'weokeuseuteisyeon'	85	17	508	211
'weokseuteisyeon'	1	1	11	9

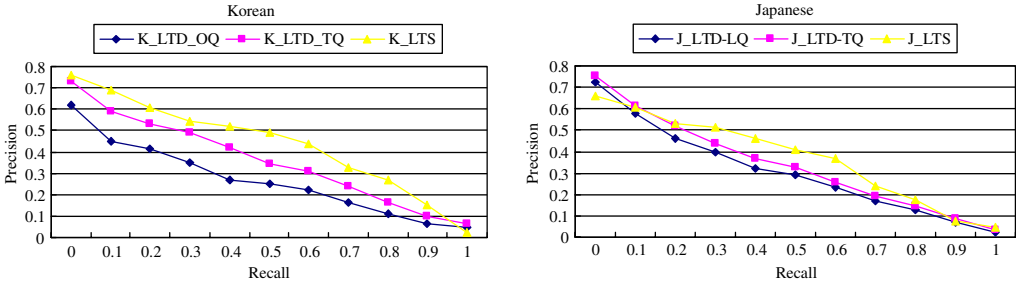

Fig. 8. Performance of LTD and LTS in Korean and Japanese.

cision of the L-Query and the T-Query is 0.1512 and 0.3949, respectively. Table 16 shows TF (Term frequency) and DF (document frequency) of the T-Query terms for relevant documents (Rel) and for all documents (All). Table 16 gives an evidence for performance improvement. 'liseukeu', which is the transliteration of 'RISC' and does not exist in L-Query, is the main factor for performance improvement because it is one of effective query terms in the T-Query for retrieving relevant documents.

Fig. 8 shows recall/precision curves for the performances of LTS queries (K_LTS: 0.4374 and J_LTS: 0.3724 in 11-pt. avg. precision) and LTD queries (K_LTD_LQ: 0.2691, K_LTD_TQ: 0.3617, J_LTD_LQ: 0.3085 and J_LTD_TQ: 0.3394 in 11-pt. avg. precision). Note that LTD_LQ and LTD_TQ mean L-Query and T-Query in LTD, respectively. For Korean, T-Query (K_LTD_TQ) shows 34.38% performance improvement compared with L-Query (K_LTD_LQ). For Japanese, T-Query (J_LTD_TQ) shows 10.0% performance improvement compared with L-Query (J_LTD_LQ).

5. Discussion

There are two research topics related to transliterations. One is transliteration pair acquisition and the other is machine transliteration (Brill, Kacmarcik, & Brockett, 2001; Collier, Kumano, & Hirakawa, 1997; Kuo & Yang, 2004; Tsuji, 2002). Machine transliteration is a task to automatically produce transliterations for given source words. The transliteration pair acquisition task is to find phonetic cognate in two languages and it is usually composed of two steps: phonetic conversion step and phonetic comparison. The phonetic conversion step phonetically converts words in one language to that in the others, like machine transliteration. Then the converted words in one language and the original words in the other language are compared in order to find the phonetic equivalence. For example, Brill et al. (2001), Tsuji (2002), & Collier et al. (1997) transform Japanese katakana words into English words with some Japanese-to-English back-transliteration rules or a back-transliteration system. Therefore, machine transliteration can serve as one of components in the transliteration pair acquisition method by offering a machine-generated transliterated form. On the other hand, results of the transliteration pair acquisition, a pair of source word and its transliterations, can be used as training data in machine transliteration. Therefore, the two research topics are complementary used for solving the problem related to transliterations.

Some machine transliteration methods have been proposed. Though they adopt the diverse array of information and mechanisms in order to produce transliterations for given source language words, they can be summarized in terms of **information type**, **information usage**, **ensemble** and **ranking** as described in Table 17.

Table 17
Comparison between ensemble-based transliteration model and the previous works

Method	Information type			Information usage		Ensemble	Ranking	
	Source grapheme	Phoneme	Correspondence	Context window	Previous output		Relevance	Text
GST	O	X	X	−1, 0	O	X	O	X
GNN	O	X	X	−1, +1	X	X	X	X
GDT	O	X	X	−3, +3	X	X	X	X
GPC	O	X	X	–	O	X	O	X
GMEM	O	X	X	−3, +3	O	X	O	X
GJSC	O	X	X	−3, 0	O	X	O	X
PWFST	X	O	X	–	–	X	O	O
PST	X	O	X	−1, 0	O	X	O	X
CRULE	O	O	O	−3, +3	O	X	X	X
HSC	O	O	X	–	–	X	O	O
HWFST	O	O	X	–	–	X	O	X
OURS	O	O	O	−3, +3	O	O	O	O

From the viewpoint of **information type**, GST (Jeong, Myaeng, Lee, & Choi, 1999; Lee, 1999; Lee & Choi, 1998), GNN (Kim, Lee, & Choi, 1999), GDT (Kang, 2001; Kang & Choi, 2000), GPC (Kang & Kim, 2000), GMEM (Goto et al., 2003), and GJSC (Li, Zhang, & Su, 2004) are **grapheme-based transliteration model** (ψ_G), while PWFST (Knight & Graehl, 1997) and PST (Lee, 1999) are **phoneme-based transliteration model** (ψ_P). The last four, CRULE (Oh & Choi, 2002), HSC (Al-Onaizan & Knight, 2002), HWFST (Bilac & Tanaka, 2004), and OURS are **grapheme- and phoneme-based transliteration model** (ψ_{GP}). Because **Correspondence**, in Table 17, represents correspondence between source grapheme and phoneme, **grapheme- and phoneme-based model** can be re-classified into **hybrid transliteration model** (ψ_H) (HSC and HWFST) and **correspondence model** (ψ_C) (CRULE and OURS) depending on whether it makes use of the correspondence or not. In the hybrid transliteration model, ψ_G and ψ_P are modeled with WFSTs or source-channel model. Then ψ_G and ψ_P are combined through the linear interpolation style. In their ψ_P, three components, source grapheme-to-phoneme probability model, phoneme-to-target grapheme probability model, and target language word probability models, are considered. In their ψ_G, only the source grapheme-to-target grapheme probability model is considered. The main disadvantage of the hybrid model is lack of consideration on dependency between source grapheme and phoneme in the combining process; while CRULE and OURS consider the dependency by using the correspondence between source grapheme and phoneme. Note that the first character in the name of each method, like **G**, **P**, **H**, and **C**, represents ψ_G, ψ_P, ψ_H, and ψ_C, respectively. As reported in Section 4.1.2 (see Table 8), ψ_{GP} rather than ψ_G and ψ_P show higher performance because ψ_{GP} can make use of both source grapheme and phoneme. Note that GMEM and GDT are indirectly compared with our transliteration method because (1) GMEM is similar to our ψ_G method based on MEM and (2) GDT is similar to our ψ_G method based on DT. In comparing ψ_C (OURS) and ψ_H (HWFST) in Table 7, ψ_C achieves better performance than ψ_H. This indicates that correspondence between source grapheme and phoneme is more useful than just combining both source grapheme and phoneme in machine transliteration.

From the viewpoint of **information usage**, a transliteration method tends to achieve better performance when it adopts wide context window and considers previous outputs. For example, GMEM and CRULE that satisfy this conditions gives more accurate results than GST, GNN, GDT, and PST which do not satisfy the condition (Goto et al., 2003; Oh & Choi, 2002). Because transliteration is sensitive to context, wider contexts give more powerful transliteration ability to machine transliteration systems. The previous works, however, limit their context window size to 3, because the context window size over 3 degrades the performance of their transliteration method (Goto et al., 2003; Kang & Choi, 2000; Oh & Choi, 2002) or does not significantly change the performance (Kang & Kim, 2000).

One transliteration model alone has limitation on reflecting dynamic behaviors of transliteration. Therefore several different transliteration models should be complementary used to achieve a high-performance machine transliteration system. The **ensemble** in Table 17 tells us whether a transliteration method uses transliteration results produced by different transliteration models. Though HSC and HWFST falling into the hybrid trans-

literation model use two different transliteration models (ψ_G and ψ_P), they use the two transliteration models for constructing their model rather than for producing transliteration results; while OURS makes use of transliteration results produced by three different transliteration model (ψ_G, ψ_P, and ψ_{GP}). For this reason, only our method uses the ensemble.

A transliteration system that gives top-*n* result is more applicable to machine translation and information retrieval, because the applications need transliterations ranked by relevance factors to which the transliteration system assigns. There are two ways of transliteration ranking. One is relevance-based ranking and the other is web-based ranking. The relevance-based ranking method makes use of probability or relevance score given by a transliteration model. If a transliteration model produces a transliteration from a source word with high probability or high relevance score, the rank of the transliteration will be high. Text-based ranking method can be regarded as validation of transliteration using texts (Grefenstette, 1999; Li & Grefenstette, 2005). In other words, text-based ranking method ranks transliterations by investigating how many documents or web documents contain each transliteration or how frequently each transliteration appears in texts. Therefore, a transliteration with high frequency (or document frequency) will be in high rank in text-based ranking method. In Table 17, "relevance" represents the relevance-based ranking method and "text" represents the text-based ranking method. From the viewpoint of the **ranking** method, previous works except PWFST and HSC did not consider the text-based ranking method. PWFST uses English word counts to rank transliterations and HSC uses web-based ranking. Without text-based ranking method, noisy transliterations, which are not used frequently in a target language, may not be effectively filtered out. Therefore, the text-based ranking method is necessary for a transliteration system to filter out the noisy transliterations. However, the relevance-based ranking method as well as the web-based ranking method should be considered simultaneously because a transliteration system depending on only the text-based ranking method may fail in ranking a transliteration due to absence of text containing the transliterations to be ranked. Our method considers the two ranking method at a time.

As a result, a good transliteration system should consider: (1) source grapheme and phoneme along with their correspondence simultaneously, (2) wide contexts and previous outputs, (3) dynamic behavior of transliterations, and (4) ranking methods for filtering out noisy transliterations. Our ensemble-based transliteration model satisfies the four conditions.

Though our ensemble-based transliteration model effectively produces transliterations, it shows its drawback when there are production errors (in other words, if there is no correct transliteration among transliterations produced by three transliteration models, see Section 4.1.3). This is the main reason of errors in our model. Therefore, a way of effectively producing correct transliterations to be ranked is the key to improve the performance of a transliteration system. One possible solution may be another ensemble of transliteration models or machine learning methods. For example, an ensemble of three machine-learning algorithms for the three transliteration models proposed in this paper, may be helpful to reduce the production errors.

6. Conclusion

This paper has described an ensemble-based transliteration model, which reflects dynamic behaviors and real-world usages of transliterations. The generation-and-test paradigm is applied; transliterations generated by our transliteration model are tested in IR domain.

This paper contributes to machine transliteration research in three aspects. First, this paper shows that correspondence between source grapheme and phoneme is very useful for machine transliteration. In comparison test among ψ_{GP}, ψ_G, and ψ_P in the same condition, ψ_{GP} utilizing the correspondence outperforms both ψ_G and ψ_P, which relyon either source grapheme or phoneme. Second, this paper shows that an ensemble of different transliteration models is one of ways for reflecting dynamic behaviors of transliteration. The ensemble of different transliteration models makes it possible for a machine transliteration system to produce the correct transliterations as many as possible by considering different ways of transliteration. Therefore, different transliteration models can cooperate with each other by making up for the weak points of each transliteration model through the ensemble. Finally, this paper shows that transliteration ranking based on web data and relevance scores is one of ways for filtering out noisy transliterations. Because web data reflects real-world

usage, noisy transliterations, which are not used in target language, can be filtered out. Through the three strong points, our ensemble-based transliteration model shows 78–80% word accuracy, which is about 28–59% improvements compared with the previous works.

The experiment on information retrieval shows that our transliteration model is useful for improving the performance in information retrieval system. Transliterations and transliteration variations produced by our transliteration system were evaluated on KTSET and NTCIR-1 test collection for Korean and Japanese, respectively. T-Query improves the performance of information retrieval system about 10–34% compared to the L-Query and about 30–58% compared to B-Query.

However, further research is needed in **transliteration production** and **transliteration ranking** in order to improve our ensemble-based transliteration model. First, further research about an ensemble of various transliteration models and various machine learning algorithms such as SVM (Vapnik, 1995) may be necessary to avoid errors in transliteration production. Second, a more sophisticated transliteration ranking method, which is independent of web data, may be necessary to rank transliterations.

Acknowledgement

This work was supported by the Korea Ministry of Science and Technology, the Korea Ministry of Commerce, Industry and Energy, and the Korea Science and Engineering Foundation (KOSEF).

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*.

Berger, A., Della Pietra, S., & Della Pietra, V. (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 IJC-NLP2004* (pp. 542–547).

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

Brill, E., Kacmarcik, G., & Brockett, C. (2001). Automatically harvesting Katakana–English term pairs from search engine query logs. *NLPRS 2001*, 393–399.

CMU. (1997). The Carnegie Mellon University (CMU) Pronouncing Dictionary v0.6. Available form http://www.speech.cs.cmu.edu/cgi-bin/cmudict.

Collier, N., Kumano, A., & Hirakawa, H. (1997). Acquisition of English–Japanese proper nouns from noisy-parallel newswire articles using KATAKANA matching. In *Proceedings of the natural language processing Pacific Rim symposium 1997* (pp. 309–314).

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, Ko van der, & Bosch, Antal van den (2002). Timble TiMBL: Tilburg Memory Based Learner. version 4.3, Reference Guide, ILK Technical Report 02–10.

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

Fujii, Atsushi, & Ishikawa, Tetsuya (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*.

Grefenstette, G. (1999). The WWW as a resource for example-based MT tasks. In *Proceedings of ASLIB'99 translating and the computer 21*.

Kando, Noriko, Kuriyama, K., Nozue, T., Eguchi, K., Kato H., & Hidaka, S. (1999). Overview of IR tasks. In *Proceedings of the first NTCIR workshop on research in Japanese text retrieval and term recognition*.

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, Athens, Greece*.

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 Department, Kaist.

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*.

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* (in Korean).

Knight, K., & Graehl, J. (1997). Machine transliteration. In: *Proceedings. of the 35th annual meetings of the association for computational linguistics (ACL), Madrid, Spain*.

Kuo, J.-S., & Yang, Y.-K. (2004). Generating paired transliterated-cognates using multiple pronunciation characteristics from Web Corpora. *PACLIC 18*.

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*, 523–540.

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

Lee, J. S., 1999, An English–Korean transliteration and retransliteration model for cross-lingual information retrieval. Ph.D. Thesis, Computer Science Department, Kaist.

Li, H., Zhang, M., & Su, J. (2004). A joint source-channel model for machine transliteration. *ACL 2004*, 159–166.

Li, Y., & Grefenstette, G. (2005). Translating Chinese romanized name into Chinese idiographic characters via Corpus and Web validation. *CORIA'2005 Conference*.

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

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

Miyao, Yusuke, & Tsujii, Jun'ichi. (2002). Maximum Entropy Estimation for Feature Forests. In *Proceedings of human language technology conference (HLT 2002)*.

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

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

Park, Y. C., Choi, K.-S., Kim, J. K., & Kim, Y. H. (1996). Development of the KT test collection for researchers in information retrieval. In *Proceedings of the 23th KISS spring conference* (in Korean).

Paul, O., & Callan, J. (2001). Experiments using the Lemur toolkit. In *Proceedings of the tenth text retrieval conference (TREC-10)*.

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.

Salton, G. (1989). *Automatic text processing: The transformation, analysis, and retrieval of information by computer*. Reading, PA: Addison-Wesley.

Tsuji, K. (2002). Automatic extraction of translational Japanese–KATAKANA and English word pairs from bilingual corpora. *International Journal of Computer Processing of Oriental Languages, 15*(3), 261–279.

Vapnik, V. N. (1995). *The nature of statistical learning theory*. Springer.

Zhang, Le. (2004). Maximum entropy modeling toolkit for Python and C++. Available from http://www.nlplab.cn/zhangle/.