An Ensemble of Grapheme and Phoneme for Machine Transliteration

Jong-Hoon Oh and Key-Sun Choi

Department of Computer Science, KAIST/KORTERM/BOLA,
373-1 Guseong-dong, Yuseong-gu, Daejeon, 305-701, Republic of Korea
{rovellia, kschoi}@world.kaist.ac.kr

Abstract. Machine transliteration is an automatic method to generate characters or words in one alphabetical system for the corresponding characters in another alphabetical system. There has been increasing concern on machine transliteration as an assistant of machine translation and information retrieval. Three machine transliteration models, including "grapheme-based model", "phoneme-based model", and "hybrid model", have been proposed. However, there are few works trying to make use of correspondence between source grapheme and phoneme, although the correspondence plays an important role in machine transliteration. Furthermore there are few works, which dynamically handle source grapheme and phoneme. In this paper, we propose a new transliteration model based on an ensemble of grapheme and phoneme. Our model makes use of the correspondence and dynamically uses source grapheme and phoneme. Our method shows better performance than the previous works about 15~23% in English-to-Korean transliteration and about 15~43% in English-to-Japanese transliteration.

1 Introduction

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, 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 and so on. There has been increasing concern on machine transliteration as an assistant of Machine Translation (MT) [2], [10], mono-lingual information retrieval (MLIR) [8], [11] and cross-lingual information retrieval (CLIR) [6]. In the area of MLIR and CLIR, machine transliteration bridges the gap between a transliterated localized form and its original form by generating all possible transliterated forms from each original form. Especially for CLIR, machine transliteration gives a help to query translation 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

[1] In this paper, target language transliterations are represented with their Romanization form in a quotation mark ('') .

proper names and technical terms are not registered in a translation dictionary. A machine transliteration system, therefore, may affect the performance of MT, MLIR, and CLIR system.

Three machine transliteration models have been studied: called "**grapheme[2]-based transliteration model (ψ_G)**" [7], [8], [9], [11], [12], [13], "**phoneme[3]-based transliteration model (ψ_P)**" [10], [12], and "**hybrid transliteration model (ψ_H)**" [2], [4], [12]. ψ_G and ψ_P are classified in terms of units to be transliterated. ψ_G is referred to the direct model because it directly transforms source language graphemes to target language graphemes without any phonetic knowledge of source language words. ψ_P is called the pivot model because it makes use of phonemes as a pivot during a transliteration process. Therefore ψ_P usually needs two steps; the first step is to produce phonemes from source language graphemes, and the second step is to produce target language graphemes from phonemes. ψ_H combines ψ_G and ψ_P with the linear interpolation style. Hereafter, we will use a source grapheme for a source language grapheme and a target grapheme for a target language grapheme.

Though transliteration is the phonetic process (ψ_P) rather than the orthographic one (ψ_G) [10], we should consider both source grapheme and phoneme to achieve high performance in machine transliteration because the standard transliterations are not restricted to phoneme-based transliterations[4]. However, many previous works make use of either source grapheme or phoneme. They simplify a machine transliteration problem into either ψ_G or ψ_P assuming that one of ψ_G and ψ_P is able to cover all transliteration behaviors. However, transliteration is a complex process, which does not rely on either source grapheme or phoneme. For example, the standard Korean transliterations of *amylase* and *data* are grapheme-based transliteration 'amillaaje' and phoneme-based transliteration 'deiteo', respectively. A machine transliteration model, therefore, should reflect the dynamic transliteration behaviors in order to produce the correct transliterations.

ψ_H has the limited power for producing the correct transliterations because it just combines ψ_G and ψ_P with the linear interpolation style. ψ_H does not consider correspondence between source grapheme and phoneme during the transliteration process. However the correspondence plays important roles in machine transliteration. For example, phoneme /AH/[5] produces high ambiguities since 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

[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, 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] In an English-to-Korean transliteration test set [14], we find that about 60% are phoneme-based transliterations, while about 30% are grapheme-based ones. The others are transliterations generated by combining ψ_G and ψ_P.

[5] 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.*

their Japanese counterparts). If we know the correspondence between source grapheme and phoneme in this context, then we can more easily infer the correct transliteration of /AH/, since a target grapheme of /AH/ usually depends on a source grapheme corresponding to /AH/. Korean transliterations of source grapheme *a* is various such as 'a', 'ei', 'o', 'eo' and so on. Like the previous example, correspondence makes it possible to reduce transliteration ambiguities like Table 1. In Table 1, the underlined source grapheme *a* in the example column is pronounced as the phoneme in the phoneme column. The correct Korean transliterations of source grapheme *a* can be more easily found, like in the Korean grapheme column, by means of phonemes in the phoneme column.

Table 1. 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, average, sil**a**ge
'o'	/AO/	**a**llspice, b**a**ll, ch**a**lk

In this paper, we propose a new machine transliteration model based on an ensemble of source grapheme and phoneme, symbolized as ψ_C ("**correspondence-based transliteration model**"). ψ_C has two strong points over ψ_G, ψ_P, and ψ_H. First, ψ_C can produce transliterations by considering correspondence between source grapheme and phoneme. As described above, correspondence is very useful for reducing transliteration ambiguities. From the viewpoint of reducing the ambiguities, ψ_C has an advantage over ψ_G, ψ_P, and ψ_H because ψ_C can more easily reduce the ambiguities by considering the correspondence. Second, ψ_C can dynamically handle source grapheme and phoneme according to their contexts. Because of this property, ψ_C can produce grapheme-based transliterations as well as phoneme-based transliterations. It can also produce a transliteration, where one part is a grapheme-based transliteration and the other part is a phoneme-based transliteration. For example, the Korean transliteration of *neomycin*, 'neomaisin', where 'neo' is a grapheme-based transliteration and 'maisin' is a phoneme-based transliteration.

2 Correspondence-Based Machine Transliteration Model

Correspondence-based transliteration model (ψ_C) is composed of two component functions (ψ_C: $\delta_p \times \delta_t$). In this paper, we refer to δ_p as a function for "**producing pronunciation**" and δ_t as a function for "**producing target grapheme**". First, δ_p produces pronunciation and then δ_t produces target graphemes with correspondence between source grapheme and phoneme produced by δ_p. The goal of the δ_p is to produce the most probable sequence of phonemes corresponding to source graphemes. For example, δ_p produces /B/, /AO/, /~/[6], /R/, and /D/ for each source grapheme, *b, o, a, r,* and *d* in *board* (see "The result of δ_p" in the right side of Fig 1). In this step, pronunciation is generated through two ways; **pronunciation dictionary search** and **pronunciation estimation**. A pronunciation dictionary contains the correct pronunciation corresponding to English words. Therefore, English words are first investigated whether they are registered in the dictionary otherwise their pronunciation is estimated by pronunciation estimation. The goal of δ_t is to produce the most probable sequence of target graphemes with correspondence between source grapheme and phoneme, which is the result of δ_p. For example, δ_t produces 'b', 'o', '~', '~', and 'deu' using the result of δ_p, *b-/B/, o-/AO/, a-/~/, r-/R/,* and *d-/D/* (see "The result of δ_t" in the right side of Fig 1). Finally, the target language transliteration, such as the Korean transliteration 'bodeu' for *board*, can be acquired by concatenating the sequence of target graphemes in the result of δ_t.

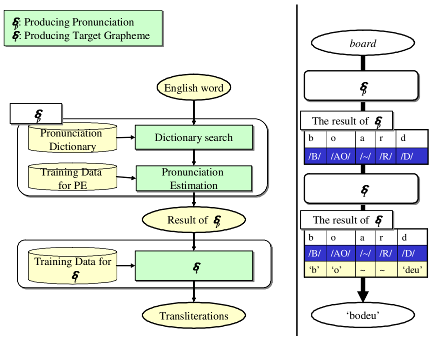


Fig. 1. The overall system architecture

Table 2. Feature types used for correspondence-based transliteration model: where *S* is a set of source graphemes (e.g. English alphabets), *P* is a set of phonemes defined in ARPABET, *T* is a set of target graphemes. Note that $f_{S,GS}$ is a symbol for indicating both f_S and f_{GS}. $f_{P,GP}$ is a symbol for indicating both f_P and f_{GP}.

Feature	Type	Description	Possible feature values
$f_{S,GS}$	f_S	Source graphemes	Source grapheme 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/, etc.)
	f_{GP}	Phoneme type	Consonant (C), Vowel (V), Semi-vowel (SV) and silence (/~/)
	f_T	Target graphemes	Target graphemes in *T*

Pronunciation estimation in δ_p and δ_t are trained by machine learning algorithms. To train each component function, we need features that represent training instance

[6] In this paper, '/~/' represents silence and '~' represents null target grapheme.

and data. Table 2 shows five feature types, f_S, f_P, f_{GS}, f_{GP}, and f_T that our model uses. Depending on component functions, different feature types are used. For example, $\delta_p(s_i)$ uses (f_S, f_{GS}, f_P) and $\delta_t(s_i, \delta_p(s_i))$ does (f_S, f_P, f_{GS}, f_{GP}, f_T).

2.1 Producing Pronunciation (δ_p)

Producing pronunciation (δ_p:$S \rightarrow P$) is a function that finds phonemes in a set P for each source grapheme, where P is a set of phonemes defined in ARPABET, and S is a set of source graphemes (e.g. English alphabets). The results of this step can be represented as a sequence of correspondences between source grapheme and phoneme. We will denote it as $GP=\{gp_1,gp_2,...,gp_n;\ gp_i=(s_i,\delta_p(s_i))\}$ where s_i is the i^{th} source grapheme of $SW=s_1,s_2,...,s_n$. Producing pronunciation is composed of two steps. The first step involves a search in the pronunciation dictionary, which contains English words and their pronunciation. This paper uses *The CMU Pronouncing Dictionary*[7], which contains 120,000 English words and their pronunciation. The second step involves pronunciation estimation. If an English word is not registered in the pronunciation dictionary, we must estimate its pronunciation.

Table 3. An example of pronunciation estimation for *b* in *board*

Feature type	L3	L2	L1	***C0***	R1	R2	R3	$\delta_p(C0)$
f_S	\$	\$	\$	***b***	o	a	r	***/B/***
f_{GS}	\$	\$	\$	***C***	V	V	C	
f_P	\$	\$	\$					

Let $SW=s_1,s_2,...,s_n$ be an English word, and $P_{SW}= p_1,p_2,...,p_n$ be SW's pronunciation, where s_i represents the i^{th} grapheme and $p_i=\delta_p(s_i)$. Pronunciation estimation is a task to find the most relevant phoneme among a set of all possible phonemes, which can be derived from source grapheme s_i. Table 3 shows an example of pronunciation estimation for *b* in *board*. In Table 3, L1~L3 and R1~R3 represent the left contexts and right contexts, respectively. C0 means the current context (or focus). δ_p(C0) means the estimated phoneme of C0. $ is a symbol for representing the start of words. The result can be interpreted as follows. The most relevant phoneme of *b*, /B/, can be produced with the context, f_S, f_{GS}, and f_P in contexts of L1~L3, C0, and R1~R3. Other phonemes for *o*, *a*, *r*, and *d* in *board* are produced in the same manner. Thus, we can get the pronunciation of *board* as /B AO R D/ by concatenating the phoneme sequence.

2.2 Producing Target Graphemes (δ_t)

Producing target graphemes (δ_t:$S\times P \rightarrow T$) is a function that finds the target grapheme in T for each gp_i that is a result of δ_p. A result of this step, GT, is represented by a sequence of gp_i and its corresponding target graphemes generated by δ_t, like $GT=\{gt_1, gt_2 ,..., gt_n;\ gt_i=(gp_i,\delta_t(gp_i))\}$.

[7] Available at http://www.speech.cs.cmu.edu/cgi-bin/cmudict

Table 4. An example of δ_t for *b* in *board*

Feature type	L3	L2	L1	***C0***	R1	R2	R3	$\boldsymbol{\delta_t(C0)}$
f_S	$	$	$	***b***	o	a	r	**‘b’**
f_P	$	$	$	***/B/***	/AO/	/~/	/R/	
f_{GS}	$	$	$	***C***	V	V	C	
f_{GP}	$	$	$	***C***	V	/~/	C	
f_T	$	$	$					

Let $SW=s_1,s_2,...,s_n$ be a source language word, $P_{SW}= p_1,p_2,...,p_n$ be SW's pronunciation and $T_{SW}= t_1, t_2,...,t_n$ be a target language word of SW, where s_i, $\delta_p(s_i)=p_i$ and $\delta_t(gp_i) = t_i$ represent the i^{th} source grapheme, phoneme corresponding to s_i, and target grapheme corresponding to gp_i, respectively. δ_t finds the most probable target grapheme among a set of all possible target graphemes, which can be derived from gp_i. δ_t produces target graphemes with source grapheme (f_S), phoneme (f_P), source grapheme type (f_{GS}), phoneme type (f_{GP}) and δ_t's previous output (f_T) in the context window. Table 4 shows an example of δ_t for *b* in *board.* δ_t produces the most probable sequence of target graphemes (e.g. Korean), like $\delta_t(gp_1)$= ‘b’, $\delta_t(gp_2)$= ‘o’, $\delta_t(gp_3)$=‘~’, $\delta_t(gp_4)$=‘~’, and $\delta_t(gp_5)$=‘deu’ for *board*. Finally, the target language transliteration of *board* as ‘bodeu’ can be acquired by concatenating the sequence of produced target graphemes.

3 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).

3.1 Maximum Entropy Model

The maximum entropy model (MEM) is a widely used probability model that can incorporate heterogeneous information effectively [3]. In the maximum entropy model, 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 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 ($fe_i(ev)=0$) [3].

δ_p and δ_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 δ_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 a index of the current source grapheme and phoneme to be transliterated and $f_{X(l,m)}$ represents features of feature type f_X located from position l to position m. Target events are a set of target graphemes (phonemes) derived from history events of δ_t (δ_p). Given history events, δ_t (δ_p) 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 history events (or activating conditions) of the feature functions by combinations of features in feature types. Possible feature combinations for history events are between features in the same feature type and between features in different feature types. The used feature combinations in each component function are listed in Table 5.

Table 5. Used feature combinations for history events

δ_p	δ_t
Between features in the same feature type	Between features in the same feature type
Between features in different feature types	Between features in different feature types
• $f_{S,GS}$ and f_P	• $f_{S,GS}$ and $f_{P,GP}$
	• $f_{S,GS}$ and f_T
	• $f_{P,GP}$ and f_T

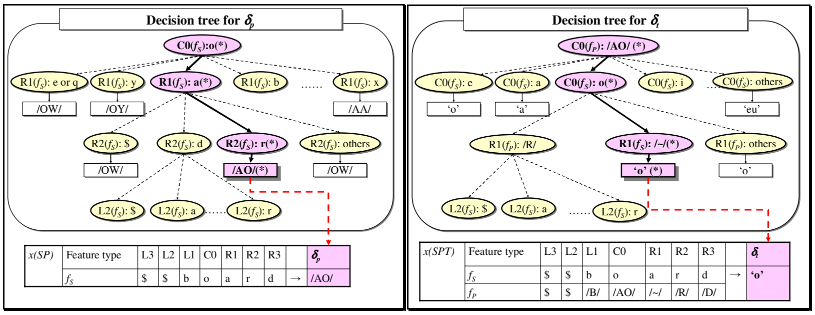
In formula (1), history events of δ_p and δ_t are defined by the conditions described in Table 5. Target events of δ_t are all possible target graphemes derived from its history events; while those of δ_p are all possible phonemes derived from its history events. In order to model each component function based on MEM, Zhang's maximum entropy modeling tool is used [16].

$$\begin{aligned} &\delta_t(s_i, \delta_p(s_i)) = \arg\max p(t_i \mid f_{T\,i-3,i-1}, f_{S,GS\,i-3,i+3}, f_{P,GP\,i-3,i+3}) \\ &\delta_p(s_i) = \arg\max p(p_i \mid f_{P\,i-3,i-1}, f_{S,GS\,i-3,i+3}) \end{aligned} \qquad (1)$$

3.2 Decision Tree

Decision tree learning is one of the most widely used and well-known methods for inductive inference [15]. ID3, which is a greedy algorithm and constructs decision trees in a top-down manner, adopts a statistical measure called *information gain* that measures how well a given feature (or attribute) separates training examples according to their target class [15]. We use C4.5 [15], which is a 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 3. Fig. 2 shows a fraction of our decision trees for δ_p and δ_t in English-to-Korean transliteration (note that the left side represents the decision tree for δ_p and the right side represents the decision tree for δ_t). A set of the target classes in the decision tree for δ_p will be a set of phonemes and that for δ_t will be a set of target graphemes. In Fig. 2, rectangles indicate a leaf node and circles indicate a decision node. In order to simplify our examples, we just use f_S and f_P in Fig. 2. Intuitively, the most effective feature for δ_p and δ_t may be located in C0 among L3~L1, C0, and R1~R3 because the correct outputs of δ_p and δ_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 δ_p and C0(f_P) for δ_t (Note that the first feature to be tested is the most effective feature). In Fig. 2, the decision tree for δ_p outputs phoneme /AO/ for the instance *x(SP)* by retrieving the decision nodes C(f_S)=*o*, R1(f_S)=*a*, and R2(f_S)=*r* represented with '*'. With the similar manner, the decision tree for δ_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 '*'.



x(SP)	Feature type	L3	L2	L1	C0	R1	R2	R3		δ_p
	f_S	\$	\$	b	o	a	r	d	→	/AO/

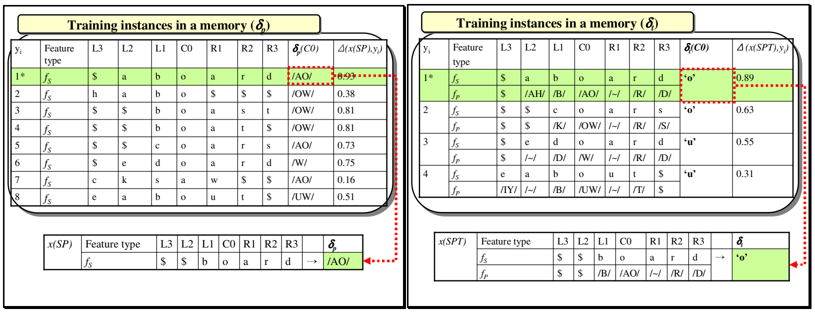
x(SPT)	Feature type	L3	L2	L1	C0	R1	R2	R3		δ_t
	f_S	\$	\$	b	o	a	r	d	→	'o'
	f_P	\$	\$	/B/	/AO/	/~/	/R/	/D/		

Fig. 2. Decision tree for δ_p and δ_t

3.3 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 [1], [5]. MBL represents a training data as a vector. In the training phase, MBL puts all training data as examples in memory, and clusters some examples with a *k-nearest neighborhood* principle. It then outputs a target class 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 assign a target class of the example to x's class. We use a memory-based learning tool called TiMBL (Tilburg memory-based learner) version 5.0 [5].

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. Fig. 3 shows examples of δ_p and δ_t based on MBL in English-to-Korean transliteration. In order to simplify our examples, we just use f_S and f_P in Fig. 3. All training data are represented with their features in the context of L3~L1, C0, and R1~R3 and their target classes for δ_p and δ_t. They are stored in the memory through a training phase. Feature weighting for dealing with features of differing importance is also performed in the training phase. In Fig. 3, δ_p based on MBL outputs the phoneme /AO/ for *x(SP)* by comparing the similarities between *x(SP)* and Y using distance metric Δ(*x(SP)*,Y). With the similar manner, δ_t based on MBL outputs the target grapheme 'o'.



Training instances in a memory (δ_p)

y_i	Feature type	L3	L2	L1	C0	R1	R2	R3	$\delta_p(C0)$	$\Delta(x(SP),y_i)$
1*	f_S	$	a	b	o	a	r	d	/AO/	0.93
2	f_S	h	a	b	o	$	$	$	/OW/	0.38
3	f_S	$	$	b	o	a	s	t	/OW/	0.81
4	f_S	$	$	b	o	a	t	$	/OW/	0.81
5	f_S	$	$	c	o	a	r	s	/AO/	0.73
6	f_S	$	e	d	o	a	r	d	/W/	0.75
7	f_S	c	k	s	a	w	$	$	/AO/	0.16
8	f_S	e	a	b	o	u	t	$	/UW/	0.51

$x(SP)$	Feature type	L3	L2	L1	C0	R1	R2	R3		δ_p
	f_S	$	$	b	o	a	r	d	→	/AO/

Training instances in a memory (δ_t)

y_i	Feature type	L3	L2	L1	C0	R1	R2	R3	$\delta_t(C0)$	$\Delta(x(SPT),y_i)$
1*	f_S	$	a	b	o	a	r	d	‘o’	0.89
	f_P	$	/AH/	/B/	/AO/	/~/	/R/	/D/		
2	f_S	$	$	c	o	a	r	s	‘o’	0.63
	f_P	$	$	/K/	/OW/	/~/	/R/	/S/		
3	f_S	$	e	d	o	a	r	d	‘u’	0.55
	f_P	$	/~/	/D/	/W/	/~/	/R/	/D/		
4	f_S	e	a	b	o	u	t	$	‘u’	0.31
	f_P	/IY/	/~/	/B/	/UW/	/~/	/T/	$		

$x(SPT)$	Feature type	L3	L2	L1	C0	R1	R2	R3		δ_t
	f_S	$	$	b	o	a	r	d	→	‘o’
	f_P	$	$	/B/	/AO/	/~/	/R/	/D/		

Fig. 3. Memory-based learning for δ_p and δ_t

4 Experiments

We perform experiments for English-to-Korean and English-to-Japanese transliteration. English-to-Korean test set (EKSet) [14] consists of 7,185 English-Korean pairs – the number of training data is 6,185 and that of test data is 1,000. EKSet contains no transliteration variations. English-to-Japanese test set (EJSet), which is an English-katakana pair in EDICT[8], consists of 10,398 – 1,000 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.) in formula (2).

$$W.A. = \frac{\#\,of\ correct\ words}{\#\,of\ \ generated\ words} \tag{2}$$

We perform two experiments called “**Comparison test**” and “**Context window size test**”. In the “Comparison test”, we compare our ψ_C with the previous works. In “Context window size test”, we evaluate the performance of our transliteration model depending on context window size.

4.1 Experimental Results

Table 6 shows results of “**Comparison test**”. MEM, DT, and MBL represent ψ_C based on maximum entropy model, decision tree, and memory-based learning, respectively. GDT [8], GPC [9], GMEM [7] and HWFST [4], which are one of the best machine transliteration methods in English-to-Korean transliteration and English-to-Japanese transliteration, are compared with ψ_C. Table 7 shows the key feature of each method in the viewpoint of **information type** (SG, PH, COR) and **information usage** (Context size, POut). Information type indicates that each transliteration method belongs to which transliteration model. For example, GDT, GPC, and GMEM will belong to ψ_G because they use only the source grapheme; while HWFST belongs to ψ_H. Information usage gives information about what kinds of information each transliteration method can deal with. From the viewpoint of information type, phoneme and correspondence, which most previous works do not consider, is the key point of the performance gap between our method and the previous works.

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

Table 6. Evaluation results of "Comparison test"

Method	EKSet		EJSet	
	W.A	Chg %	W.A	Chg %
GDT	51.4%	23.2%	50.3%	43.5%
GPC	55.1%	17.6%	53.2%	35.7%
GMEM	55.9%	16.4%	56.2%	28.5%
HWFST	58.3%	14.7%	62.5%	15.5%
DT	62.0%	7.3%	66.8%	8.1%
MEM	63.3%	5.4%	67.0%	7.8%
MBL	66.9%	0%	72.2%	0%

Table 7. Key features of our machine transliteration model and the previous works: SG, PH, COR and POut represent source grapheme, phoneme, correspondence and previous output, respectively

Method	SG	PH	COR	Context size	POut
GDT	O	X	X	<-3, +3>	X
GPC	O	X	X	Unbounded	O
GMEM	O	X	X	<-3, +3>	O
HWFST	O	O	X	-	-
Ours	O	O	O	<-3, +3>	O

From the viewpoint of information usage, if a transliteration model adopts wide context window and considers previous outputs, it tends to show better performance. For example, GMEM that satisfies the conditions gives more accurate results than GDT which does not satisfy one of them. Because machine transliteration is sensitive to context, wider contexts give more powerful transliteration ability to machine transliteration systems. Note that the previous works, however, limit their context window size to 3, because the context window size over 3 degrades the performance [8] or does not change the performance of their transliteration model [9]. Determining reasonable context window size, therefore, is very important for machine transliteration.

For "**Context window size test**", we use ψ_C based on MBL, which shows the best performance among three machine learning algorithms in Table 6. Experiments are performed by changing the context window size from 1 to 5. Table 8 shows results of context window size test. The results indicate 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 information. For example, in order to produce the correct target grapheme of *t* in *-tion*, we need the right three graphemes of *t*, *-ion*. When the context window size is over 3, it is difficult to generalize the training data because of increase of variety of the training data. With the two reasons, our system shows the best performance when the context window size is 3. Table 8 also shows that context size should be at least 2 to avoid significant decrease of performance due to lack of contextual information.

Table 8. Evaluation results of "Context window size test"

Context Size	EKSet	EJSet
1	54.5%	62.7%
2	63.3%	70.0%
3	66.9%	72.2%
4	63.9%	70.7%
5	63.8%	69.3%

In summary, our method shows significant performance improvement, about 15%~23%, in English-to-Korean transliteration, and about 15%~ 43% in English-to-Japanese transliteration. Experiments show that a good transliteration system should consider; 1) source grapheme and phoneme along with their correspondence simultaneously and 2) reasonable context size and previous output. Our transliteration model satisfies the two conditions, thus it shows higher performance than the previous works.

5 Conclusion

This paper has described a correspondence-based machine transliteration model (ψ_C). Unlike the previous transliteration models, ψ_C uses correspondence between source grapheme and phoneme. The correspondence makes it possible for ψ_C to effectively produce both grapheme-based transliterations and phoneme-based transliterations. Moreover, the correspondence helps ψ_C to reduce transliteration ambiguities more easily. Experiments show that ψ_C is more powerful transliteration model than the previous transliteration models (ψ_C shows significant performance improvement, about 15%~23%, in English-to-Korean transliteration, and about 15%~ 43% in English-to-Japanese transliteration).

In future work, we will apply our transliteration model to English-to-Chinese transliteration model. In order to prove usefulness of our method in NLP applications, we need to apply our system to applications such as automatic bi-lingual dictionary construction, information retrieval, machine translation, speech recognition and so on.

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

1. Aha, D. W. Lazy learning: Special issue editorial. Artificial Intelligence Review, 11:710, (1997).
2. Al-Onaizan Y. and Kevin Knight, "Translating Named Entities Using Monolingual and Bilingual Resources", In the Proceedings of ACL 2002, (2002)

3. Berger, A., S. Della Pietra, and V. Della Pietra. , A maximum entropy approach to natural language processing. Computational Linguistics, 22(1), (1996), 39—71
4. Bilac Slaven and Hozumi Tanaka. "Improving Back-Transliteration by Combining Information Sources". In Proc. of IJC-NLP2004, (2004) 542—547
5. Daelemans, W., Jakub Zavrel, Ko van der Sloot, and Antal van den Bosch, 2002, Timble TiMBL: Tilburg Memory Based Learner, version 4.3, Reference Guide, ILK Technical Report 02-10, (2002).
6. Fujii, Atsushi and Tetsuya, Ishikawa. Japanese/English Cross-Language Information Retrieval: Exploration of Query Translation and Transliteration. Computers and the Humanities, Vol.35, No.4, (2001) 389—420
7. Goto, I., N. Kato, N. Uratani and T. Ehara, Transliteration Considering Context Information Based on the Maximum Entropy Method, In Proceedings of MT-Summit IX, (2003)
8. Kang B.J. and K-S. Choi, "Automatic Transliteration and Back-transliteration by Decision Tree Learning", In Proceedings of the 2nd International Conference on Language Resources and Evaluation, (2000)
9. Kang, I.H. and G.C. Kim, "English-to-Korean Transliteration using Multiple Unbounded Overlapping Phoneme Chunks", In Proceedings of the 18th International Conference on Computational Linguistics, (2000).
10. Knight, K. and J. Graehl, "Machine Transliteration". In Proceedings. of the 35th Annual Meetings of the Association for Computational Linguistics (ACL), (1997)
11. Lee, J. S. and K. S. Choi, English to Korean Statistical transliteration for information retrieval. Computer Processing of Oriental Languages, 12(1), (1998), 17-37.
12. Lee, J.S., An English-Korean transliteration and Retransliteration model for Cross-lingual information retrieval, PhD Thesis, Computer Science Dept., KAIST, (1999)
13. Li Haizhou, Min Zhang and Jian Su , A Joint Source-Channel Model for Machine Transliteration , ACL 2004, (2004), 159—166
14. Nam, Y.S., Foreign dictionary, Sung-An-Dang publisher, (1997)
15. Quinlan, J.R., "C4.5: Programs for Machine Learning", Morgan Kauffman, (1993)
16. Zhang, Le. Maximum Entropy Modeling Toolkit for Python and C++. http://www.nlplab.cn/zhangle/, (2004)