Automatic Transliteration and Back-Transliteration by Decision Tree Learning

Byung-Ju Kang, Key-Sun Choi

Department of Computer Science
Advanced Information Technology Research Center (AITrc)
Korea Terminology Center for Language and Knowledge Engineering
Korea Advanced Institute of Science and Technology
373-1 Kusong-dong, Yusong-gu, Taejon, 305-701, Korea
{bjkang, kschoi}@world.kaist.ac.kr

Abstract

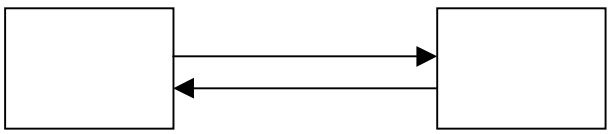
Automatic transliteration and back-transliteration across languages with drastically different alphabets and phonemes inventories such as English/Korean, English/Japanese, English/Arabic, English/Chinese, etc, have practical importance in machine translation, cross-lingual information retrieval, and automatic bilingual dictionary compilation, etc. In this paper, a bi-directional and to some extent language independent methodology for English/Korean transliteration and back-transliteration is described. Our method is composed of character alignment and decision tree learning. We induce transliteration rules for each English alphabet and back-transliteration rules for each Korean alphabet. For the training of decision trees we need a large labeled examples of transliteration and back-transliteration. However this kind of resources are generally not available. Our character alignment algorithm is capable of highly accurately aligning English word and Korean transliteration in a desired way.

1. Introduction

Recently there are increasing concern about automatic transliteration and back-transliteration across languages, especially with radical differences in their alphabets and phoneme inventories such as English/Korean (Lee & Choi, 1998; Jeong et al., 1999), English/Japanese (Knight & Graehl, 1997), English/Arabic (Stalls & Knight, 1998), English/Chinese (Wan & Verspoor, 1998), etc. Transliteration is phonetic translation that finds the phonetic equivalent in target language given a source language word. Back-transliteration is the backward process that finds the origin word from the transliterated word. For example, English word "internet" is generally transliterated into '인터넷 (intonet)' in Korean and the right back-transliteration of '인터넷 (intonet)' should be 'internet' (Figure 1).

Most of the related researches have been done under the context of cross-lingual information retrieval (Lee & Choi, 1998), machine translation (Knight & Graehl, 1997), automatic bilingual dictionary compilation (Kang & Maciejewski, 1996; Collier et al., 1997) and resolution of the word mismatch problem caused by foreign words in information retrieval (Jeong et al., 1999).

In contemporary Korean, the significant portion of the vocabulary consists of loan words and most of them are from English. They are generally transliterated in Korean. Both in machine translation and cross-lingual information retrieval, source language word should be translated into target language word based on bilingual dictionary. Unfortunately, many English technical terms or names are not listed in the bilingual dictionary. Therefore, when source language is English and target language is Korean, automatic transliteration is required (Lee & Choi, 1998), on the contrary, when source language is Korean and target language is English, automatic back-transliteration is required (Lee, 1999).



Recently, in Korean, the speed of import of English words is accelerated. Most of them are technical terms. We believe that this phenomenon is also true in most of languages besides English. It is difficult to expect all the new imported words be immediately listed in contemporary Korean dictionary. However NLP (Natural Language Processing) can not avoid processing of the new words under the excuse of the non-existence in dictionary. It is possible to automatically compile foreign word dictionary by phonetic matching among English words and transliterations in parallel or non-parallel bilingual corpora (Kang & Maciejewski, 1996; Collier et al., 1997).

In Korean, transliteration of an English word may be very various. For example, English word 'digital' may be variously transliterated in Korean as '디지털 (ticithel)', '디지탈 (ticithal)', and '디지틀 (ticithul)', etc, even though '디지털 (ticithel)' is preferred as a standard form. This is because an English phoneme can only be ambiguously mapped to more than one Korean phoneme due to their radically different phonolgies. Moreover, lazy writers often use English word in its original form without transliteration. These mixed use of various transliterations together with their origin English word cause severe word mismatch problem in information retrieval. One solution for this problem is to index only the canonical form of the variations. As the canonical form, the origin English word should be preferred. Automatic back-transliteration is required to find the canonical form of the variants (Jeong et al., 1999).

For the machine learning of transliteration and back-transliteration rules, large phonetically aligned pairs of English word and Korean transliteration are essential.

However this kind of resources does not exist and manual alignment would be highly tedious. In previous researches small amount of aligned word pairs were manually constructed (Jeong et al, 1999) or unsupervised learning method, a variation of EM(Expectation Maximization) algorithm, were tried (Lee & Choi, 1998). However manual alignment is not proper for the large amount of data and the performance of unsupervised learning was not good enough.

In this paper, we first present an automatic character alignment method between English word and Korean transliteration. Our method is highly accurate so that more than 99% accuracy was obtained. With this alignment method the construction of large reliable labeled (aligned) examples is just instant.

Once we have enough aligned data, in fact any supervised learning method would be applicable to the transliteration and back-transliteration domain. We chose decision tree method to automatically induce transliteration and back-transliteration rules for its conceptual simplicity and empirically verified effectiveness in text-to-speech domain (Dietterich et al, 1995).

Our methodology is fully bi-directional, i.e. the same methodology is used for both transliteration and back-transliteration. Moreover, our proposed methodology is easily adaptable to other language pairs. Decision tree learning algorithm is domain independent and our alignment method is relatively easily adaptable to other language pairs. The alignment method is logically divided into language independent part and language dependent part. Alignment algorithm itself is domain independent but the heuristic rules used incorporate bilingual phonemic knowledge. However the bilingual phonemic knowledge is simple so that even non-expert may encode it without much difficulty.

2. Character Alignment

2.1. English/Korean character alignment

English/Korean character alignment is, given a source language word (English) and its phonetic equivalent in target language (Korean), to find the most phonetically probable correspondence between their characters. For example, English word 'board' is generally transliterated into '보드 (potu)'[1] in Korean and their one possible alignment is as follows:

English	b	oa	r	d
	|	|	|	|
Korean	ㅂ	ㅗ	-	ㄷㅡ

Generally E/K character alignment has the following properties:

(1) many-to-many correspondence
(2) null correspondence
(3) no crossing dependency

In most cases English words and their Korean transliterations do not have same length. This means that the mapping type is generally many-to-many correspondences. Moreover it may also have null correspondence. They are mainly due to silent English letters. They are also caused by Korean characters. In Korean a consonant is not allowed come in stand-alone and must be followed by a vowel. This causes dummy Korean vowels that can not be mapped to any English alphabets. In character alignment, unlike word alignment and sentence alignment, crossing dependency does not occur. In another words, when the correspondences are visualized as links between English and Korean characters, the links do not cross each other. No crossing dependency property makes an effective alignment algorithm possible by drastically reducing the search space.

Let's call the mapping unit, 'b', 'oa', 'r', 'd', 'ㅂ', 'ㅗ' 'ㄷㅡ' in the above alignment example, as PU (Pronunciation Unit) (Lee & Choi, 1998). We may use decision trees for the induction of the mapping rules between English PUs and Korean PUs. Unfortunately, however, too many PUs may be produced and consequently too many decision trees need to be constructed. Moreover null PUs in the source word side makes the application of decision tree method difficult. To remedy this problem we constrain the alignment configuration. Specifically we allow only one-to-many correspondence and prohibit null PUs in the source word side.

Under these constraints the previous alignment example may be modified as follows:

English	b	o	a	r	d
	|	|	|	|	|
Korean	ㅂ	ㅗ	-	-	ㄷㅡ

On the contrary, when source word is Korean and target word is English, i.e. back-transliteration, the alignment should be as follows:

Korean	ㅂ	ㅗ	ㄷ	ㅡ
	|	|	|	|
English	b	o a r	d	-

This constrained version of character alignment makes decision tree learning more manageable and more efficient. In the case of E/K transliteration only 26 decision trees for each English alphabet need to be learned and in the case of E/K back-transliteration only 46[2] decision trees for each Korean alphabet need to be learned.

2.2. Alignment algorithm

For the automatic character alignment, we developed an extended version of Covington's alignment algorithm (Covington, 1996). Covington's algorithm views an alignment as a way of stepping through two words while performing *match* or *skip* operation on each step. Thus the alignment

[1] Korean characters are composed in syllable unit when they get written. The two-syllable word '보드' may be deformed as 'ㅂㅗㄷㅡ' in character unit.

[2] Refer to appendix for the more information about the 46 Korean alphabets.

source	b	o	a	r	d	-
target	ㅂ	ㅗ	-	-	ㄷ	ㅡ

is produced by matching 'b' and 'ㅂ', 'o' and 'ㅗ', then skipping 'a' and 'r', matching 'd' and 'ㄷ', and lastly skipping 'ㅡ'. Null symbol '-' indicates skip at the position.

Covington's algorithm produces only one-to-one correspondence. This implies that null mapping is inevitable on both source and target word side. In order to produce one-to-many correspondences and remove null on the source word side we introduce *bind* operation. We define two kinds of bind operation: *forward bind* and *backward bind*. The following alignment example of English word 'switch' and Korean transliteration '스위치[3] (suwuichi)' pictorially represents the two bind operations.

source	ㅅ	ㅡ	>	ㅟ	>	ㅊ	<	ㅣ
target	s	-	w	i	t	c	h	-

where '>' and '<' respectively represent forward bind and backward bind at the position. 'w' is forward binded with 'i' and together matched with 'ㅟ'. Similarly, 't' and 'h' is forward and backward binded with 'c' and collectively matched with 'ㅊ'. By introducing bind operations we can remove null on the source side. Therefore the recurrent alignment example of 'board' and '보드 (potu)' may be represented as follows:

source	b	o	a	r	d	<
target	ㅂ	ㅗ	-	-	ㄷ	ㅡ

in the case of transliteration and

source	ㅂ	ㅗ	<	<	ㄷ	ㅡ
target	b	o	a	r	d	-

in the case of back-transliteration.

We can systematically generate all the valid alignments that are possible by match, skip, bind operations and satisfy the alignment constraints. Aligning may be interpreted as finding the best alignment in the alignment search space that is composed of all the valid alignments. The algorithm does not use dynamic programming but does depth-first search while pruning fruitless branches early (Covington, 1996). To evaluate each alignment, every match, skip, bind is assigned a penalty depending on the phonetic similarity between the English letter and Korean character under consideration. The alignment that has the least total penalty summed over all the operations is determined as the best. For example, the total penalty of the alignment of 'board' and '보드 (potu)' can be computed as follows:

English	b	o	a	r	d	<	
Korean	ㅂ	ㅗ	-	-	ㄷ	ㅡ	
operation	M	M	S	S	M	b.B	
penalty	0	+ 10	+ 40	+ 60	+ 0	+ 200	= 310

[3] '스위치' is deformed as 'ㅅㅡㅟㅊㅣ' where 'ㅇ' is omitted since it is soundless when it is syllable-initial.

Human who has a little bit of bilingual phonemic knowledge can almost correctly align any English word and its Korean transliteration pair. This is because relatively simple bilingual phonemic knowledge is sufficient for the alignment task. We hope to simulate this human process. We may exploit the following two heuristics that are expected to be very effective in E/K character alignment.

H1. Consonant tends to map with consonant and vowel tends to map with vowel.
H2. There exist typical Korean transliterations for each English alphabet.

We have succeeded in aligning with high accuracy using the heuristic H1 and H2. The heuristic H1 seems to always hold except 'w'. The semi-vowel 'w' is sometimes mapped to Korean consonant even though it is usually mapped to vowels. For the heuristic H2, we can easily make a list of typical Korean transliterations for each English alphabet. Generally an English alphabet has more than one Korean character that is phonetically similar. Table 1 lists phonetically similar Korean transliterations (or characters) for several English alphabets. This simple bilingual phonemic knowledge can be coded without much effort by even non-expert. If we match English alphabet with the Korean character in the list with higher priority, in most cases we get correct alignment. To handle more complicated cases we made up of the evaluation metrics in Table 2 by extending the heuristic H1.

The evaluation metrics in Table 2 approximately realize the following principles: (1) phonetic similarity matters most, (2) consonants matters more than vowel, (3) vowel and consonant do not tend to map each other, (4) phonetically dissimilar consonants do not tend to map each other, rather skip is preferred, (5) phonetically dissimilar vowels tend to map each other rather than skipping, (6) bind is more generous for dissimilar consonants' matching and also for vowel/consonant matching, (7) consonant/vowel bind preferred than dissimilar consonant bind.

We ran our alignment algorithm on 500 pairs[5] of English word and Korean transliteration. The alignment results were manually checked by human evaluators.

English alphabet	Korean transliterations
a	ㅏ ㅐ ㅑ ㅓ ㅔ ㅗㅗ
b	ㅂ ㅂ*[4]
d	ㄷ ㄸ
o	ㅗ ㅏ
r	ㄹ ㄹ*

Table 1. Typical Korean transliterations for several English alphabets

[4] The consonant attached with '*' means that it is a syllable-final consonant. Refer to appendix for more explanation.

[5] The 500 word pairs were randomly selected from the 7000 word pairs that are explained in section 4.1.

operation	condition	penalty
match	similar consonant / consonant	0
	similar vowel / vowel	10
	dissimilar vowel / vowel	30
	dissimilar consonant / consonant	240
	vowel / consonant	250
skip	vowel	40
	consonant	60
bind	similar consonant / consonant	0
	similar vowel / vowel	10
	dissimilar vowel / vowel	30
	dissimilar consonant / consonant	190
	vowel / consonant	200

Table 2. Alignment evaluation metrics

Table 3 shows the percent correct when alignment direction is from English to Korean (for transliteration) and from Korean to English. (for back-transliteration). In both cases extremely high performances, more than 99% accuracy, were obtained.

alignment direction	percent correct
English → Korean	99.4%
Korean → English	99.0%

Table 3. The performance of the character alignment algorithm

3. Learning decision trees

Once aligned English word - Korean transliteration pairs are prepared, it is very straightforward to generate large training data for the decision tree induction. For the automatic transliteration (from English to Korean) the following five mapping examples may be obtained from the constrained alignment of 'board' and '보드 (potu)' .

L3	L2	L1	(E)	R1	R2	R3		E
<	<	<	(b)	o	a	r	→	ㅂ
<	<	b	(o)	a	r	d	→	ㅗ
<	b	o	(a)	r	d	>	→	-
b	o	a	(r)	r	>	>	→	-
o	a	r	(d)	>	>	>	→	ㄷㅡ

Each example consists of 6 attribute values, left three characters and right three characters and is labeled with the corresponding Korean transliteration. These labeled examples are classified by English alphabet and then used as training data for the learning of 26 decision trees.

On the other hand, for the back-transliteration (from Korean to English) the following four mapping examples may be obtained.

L3	L2	L1	(K)	R1	R2	R3		E
<	<	<	(ㅂ)	ㅗ	ㄷ	ㅡ	→	b
<	<	ㅂ	(ㅗ)	ㄷ	ㅡ	>	→	oar
<	ㅂ	ㅗ	(ㄷ)	ㅡ	>	>	→	d
ㅂ	ㅗ	ㄷ	(ㅡ)	>	>	>	→	-

These examples are classified by Korean alphabet and then used as training data for the learning of 46 decision trees.

We use ID3-like algorithm for the learning of the decision trees. ID3 is a simple decision tree learning algorithm developed by Quinlan (1986). ID3 constructs a decision trees recursively starting at the root node. At each node an attribute is selected and tested, then examples are partitioned depending on the values of the attribute. If all the examples of a node belong to the same class, the node become a leaf and labeled with the class. If there is no more attributes remained to test, then the node become a leaf and labeled with the majority class of the examples of the node.

Once the decision trees are independently learned, the transliteration process is straightforward. Given an input English word, each English letter is mapped to Korean characters using the corresponding decision trees, then concatenating all the Korean characters produces final Korean transliteration. However, the simple concatenation of the Korean characters may not result in a valid Korean word. This is because there exist explicit rules for the composition of a valid syllable from Korean characters. Currently we simply add Korean vowel 'ㅡ'[6] in default if it is needed to compose a valid syllable. However more elaborate method would be desirable. Back-transliteration can be done in similar way.

4. Experiments

4.1. Training and testing data

7,000 foreign word and Korean transliteration pairs were prepared as experiment data. The 7,000 transliteration examples were selected from the foreign word dictionary of Nam (1977), which contains about 9,000 standard transliterations with their origin word, after discarding acronyms, multi-words, and words containing non-English alphabets, etc. However, the data still contains various non-English origin words like French, Russian, Spanish, and Italian, etc, which would degrade the learning performance. Most of words in the data are from English, however, so non-English origin words would act as noise during the learning process.

1000 words, out of the 7,000 words, were randomly chosen and reserved as testing data. The remained 6,000 words were equally divided into 3000 word training data set and 3000 word extra data set. For all the following experiments in this paper the 1000-word testing data and 3000-word training data were used in default. And the 3000-word extra data were used as validation data or extra training data depending on the kind of experiment.

[6] The Korean vowel 'ㅡ' consumes the least energy to pronounce. So it tends to be easily attached to a standalone consonant than any other vowels.

attribute selection method	transliteration				back-transliteration			
	word accuracy	character accuracy	letter accuracy	Total no. of leaves	word accuracy	character accuracy	letter accuracy	Total no. of leaves
info. gain	44.9	80.5	84.9	7363	34.2	78.2	81.5	7767
proximity	48.7	81.8	86.3	6973	34.7	78.6	82.1	7738

Table 4. Performance comparison between two attribute selection methods: proximity and information gain.

4.2. Evaluation measures

Transliteration and back-transliteration accuracy is measured by the percentage of the number of correctly transliterated or back-transliterated words divided by the total number of words tested. The correct transliteration here means the standard transliteration as listed in the dictionary and the correct back-transliteration means the origin English word. This measure is called *word accuracy*.

However word accuracy has some problems as a transliteration accuracy measure. There are generally several acceptable transliterations, though they are not all standard, for an English word. The decision about the acceptability is more or less subjective to human evaluator. In order to alleviate the problem, we introduce *character accuracy* (Lee & Choi, 1998), which is equivalent to the minimum edit distance measure (Wagner & Fischer, 1974) for string comparison, that computes the least number of character insertions, deletions, substitutions needed to transform a generated word to its correct form.

We use another measure, called *letter accuracy*, which is the number of correctly transliterated or back-transliterated letters divided by the total number of letters in the test set. Using the letter accuracy we can evaluate how well an individual decision tree has been learned for each English alphabet or Korean alphabet.

The word, letter, and character accuracy are defined as follows:

$$\text{word accuracy} = \frac{\text{no. of correctly (back-)transliterated words}}{\text{total number of words in test set}}$$

$$\text{letter accuracy} = \frac{\text{no. of correctly (back-)transliterated letters}}{\text{total number of letters in test set}}$$

$$\text{character accuracy} = \frac{L-(i+d+s)}{L}$$

where L is the length of the original string, and i, d, and s are the number of insertion, deletion and substitution respectively. If the dividend is negative (when $L < (i + d + s)$), we consider it zero. The term character accuracy is actually used as the meaning of average character accuracy of all the tested words.

4.3. Attribute selection

ID3 algorithm uses information gain (Quinlan, 1986) as attribute selection measure. Information gain measure is knowledge-poor approach based solely on the statistics on the training data. In our case, the training data may also contain noises and many irregularities. Consequently, it is inherently impossible to determine the perfect ordering of attribute selection only by seeing the training data.

In this vein, we tested alternative orderings of attribute testing based on the domain knowledge about E/K transliteration. The domain knowledge we exploited is the following two hypothesis. The first is that nearer an attribute alphabet is to the target alphabet, more influence it gives to the transliteration of the target alphabet. The second is that each alphabet should be differentiated in considering their transliteration context.

The first hypothesis is quite obvious. For the second hypothesis, let's consider the case of 'h', alphabet 'h' is very often combined with other consonants so as to form a single phoneme like 'ch', 'th', and 'sh', etc. Therefore the immediately preceding alphabet is more likely to dictate the transliteration of 'h'. Based on these two hypothesis or observations, we devised two alternative orderings of attribute testing for each alphabet: *right-first proximity* and *left-first proximity* selection strategy selects attributes in the order of <R1, L1, R2, L2, R3, L3> and <L1, R1, L2, R2, L3, R3> respectively.

Our attribute selection strategy is to select nearer attribute first and to determine the preferred direction (right-first or left-first) as that of high accuracy in validation data that are set aside from training data. Table 4 shows the performance comparison between our proximity selection method and information gain measure with the default 3000-word training data and 1000-word testing data. As validation data for proximity method the extra data set of 3000 words was used. The proximity method performed better in both transliteration and back-transliteration. The effectiveness was much more obvious in transliteration but the performance difference in back-transliteration looks trivial. In all the following experiments, the proximity attribute ordering is used in default unless other attribute selection measure is explicitly specified.

4.4. Pruning trees

ID3 algorithm grows the tree until all the training examples are perfectly classified. If the training data contains noise, the pure ID3 algorithm may lead to some difficulty. This is because decision tree tries to learn not only the target concept but also the peculiar pattern of the noise. So, even though the decision tree may almost perfectly perform on the training data, it will perform very poorly on unseen data. This phenomenon is called *overfitting* (Mitchell, 1997).

pruning method	transliteration				back-transliteration			
	word accuracy	character accuracy	letter accuracy	Total no. of leaves	word accuracy	character accuracy	letter accuracy	Total no. of leaves
before prun	48.7	81.8	86.3	6973	34.7	78.6	82.1	7738
pre-prun*	48.8	82.0	86.4	6723	34.7	79.1	82.4	5513
post-prun**	48.5	81.7	86.3	6937	32.3	79.3	82.3	4123

* using information gain measure, ** using reduced error pruning

Table 5. Effectiveness of tree pruning

training data size	transliteration				back-transliteration			
	word accuracy	character accuracy	letter accuracy	Total no. of leaves	word accuracy	character accuracy	letter accuracy	Total no. of leaves
1000	38.9	76.8	82.4	2950	27.3	75.2	79.2	3104
2000	45.0	80.5	85.1	5017	31.7	77.6	81.2	5565
3000	48.7	81.8	86.3	6973	34.7	78.6	82.1	7738
4000	49.7	82.8	86.9	9922	36.4	79.2	82.6	10646
5000	50.9	83.1	87.3	12355	35.9	79.5	83.0	13167
6000	51.3	83.4	87.5	14547	37.2	80.0	83.3	15423

Table 6. Effectiveness of enlarging the training data size.

There are two fundamentally different approaches to avoiding overfitting the training data in decision tree learning. First approach, called pre-pruning, is to stop growing decision trees before it reaches to the perfect classification of all the examples. Second approach, called post-pruning, is to derive complete decision tree first, then to post-prune the tree.

The most critical part of pre-pruning is when to stop growing. Many stopping criterions have been proposed so far (Mingers, 1989). But we tested only one of them in our experiments. The information gain measure was adopted since we did not use it as a selection measure[7]. If the information gain acquired by growing the current node is less than a specified threshold, stops growing and make the current node as a leaf node. The majority class of the examples of the node is assigned as the class of the leaf node.

Generally, post-pruning is known to be more effective method to avoiding overfitting problems than pre-pruning (Mitchell, 1997). So we tested *reduced error pruning* method (Quinlan, 1987), which is one of the most representative post-pruning methods and known to give persistent performance (Mingers, 1989), to find out whether it can improve the transliteration accuracy.

In the past work on English text-to-speech mapping, pruning trees turned out to be not much helpful in improving the mapping accuracy (Dietterich et al, 1995). We also obtained similar results. All the tree pruning methods failed to increase both the transliteration and back-transliteration accuracy (Table 5). This result may mean that ID3 is not overfitting the data. In another words, the 3000-word data does not contain much noise. This result is not so surprising since our character alignment is almost perfect and the only source of the noise is the relatively small amount of non-English origin words.

Another interesting observation is that in back-transliteration, both pre-pruning and post-pruning reduced the tree sizes by 28% and 46% respectively without decreasing back-transliteration accuracy while no significant change in tree size happened in transliteration.

4.5. Training set size

Until now, we have only showed performance values with the 3000-word training set. It might be that more training data raise the transliteration accuracy. So we generated decision trees, increasing data by 1000 words, with 6 different training data of the size of 1000 words to 6000 words (Table 6). We obtained significant performance improvement especially in word accuracy. Transliteration word accuracy increased from 38.9 (1000 words) to 51.3 (6000 words) by about 44% and back-transliteration word accuracy increased from 27.3 (1000 words) to 37.2 (6000 words) by about 36%. However, in both transliteration and back-transliteration, the increase rates were rapidly slowed after 3000 words.

4.6. Discussions

We analyzed how well individual decision trees are learned. In the case of transliteration, it was found that all the English vowels ‘a’, ‘e’, ‘i’, ‘o’, ‘u’ and consonant ‘w’, ‘x’ had especially low letter accuracy. The low accuracy of vowel is because it is realized to many different phonemes. In the case of back-transliteration, generally Korean vowels had relatively low accuracy in the same reason.

The low letter accuracy might mean that their transliteration or back-transliteration is very irregular or their target concepts may be very complex. In the case that

[7] If information gain measure is used as an attribute selection measure, generally chi-square test is used as the stopping criterion. Using the same measure for both attribute selection and pre-pruning may cause problem (Quinlan, 1986).

transliteration and back-transliteration is inherently highly irregular, there is no good way to improve the accuracy. However, if the low accuracy is due to the high complexity of the target concept, more training data or wider context window might improve the accuracy.

5. Conclusion

We have presented a very effective bi-directional automatic English/Korean transliteration and back-transliteration methodology. Our method consists of character alignment and decision tree learning. We wanted to induce the transliteration rules for each English alphabet and the back-transliteration rules for each Korean alphabet. However what is missing is large labeled examples for the training of decision trees. So, we developed a highly accurate character alignment algorithm, which is able to align two words in a desirable constrained way across languages. Our method is somewhat language independent. The only language dependent part is the alignment evaluation metrics that may also be easily constructed without much effort.

Appendix: Korean consonants and vowels

Korean alphabets consist of 19 consonants and 21 vowels. Korean characters are composed in syllable unit when they get written. The possible syllable configurations are CV, CVC and CVCC. The consonant preceding vowel is called syllable-initial and the consonants following vowel are called syllable-finals. For phonetic matching, it is beneficial to differentiate Korean consonants as syllable-initials and syllable-finals since they are pronounced differently depending on their position within a syllable.

Table 7 shows the list of Korean consonants and vowels. In Korean transliteration of foreign words only 7 consonants are allowed in syllable-final position. The syllable-initial 'ㅇ' is dummy for the purpose of just satisfying the legitimate Korean syllable configurations. So we decided to omit it in our deformed form of Korean words.

		Korean alphabets
consonant	syllable-initial (19)	ㄱ, ㄲ, ㄴ, ㄷ, ㄸ, ㄹ, ㅁ, ㅂ, ㅃ, ㅅ, ㅆ, (ㅇ), ㅈ, ㅉ, ㅊ, ㅋ, ㅌ, ㅍ, ㅎ
	syllable-final (7)	ㄱ*, ㄴ*, ㄹ*, ㅁ*, ㅂ*, ㅅ*, ㅇ*
vowel	(21)	ㅏ, ㅐ, ㅑ, ㅒ, ㅓ, ㅔ, ㅕ, ㅖ, ㅗ, ㅘ, ㅙ, ㅚ, ㅛ, ㅜ, ㅝ, ㅞ, ㅟ, ㅠ, ㅡ, ㅢ, ㅣ

Table 7. Korean consonant and vowels

Acknowledgements

This work was supported by the Korea Science and Engineering Foundation (KOSEF) through the Advanced Information Technology Research Center (AITrc).

References

Collier, N., A. Kumano and H. Hirakawa, 1997. Acquisition of English-Japanese proper nouns from noisy-parallel newswire articles using Katakana matching. *Natural Language Processing Pacific Rim Symposium '97.*

Covington, M. A., 1996. An algorithm to align words for historical comparison. *Computational Linguistics*, 22.

Dietterich, T. G., H. Hild and G. Bakri, 1995. A Comparison of ID3 and Backpropagation for English Text-to-Speech Mapping. *Machine Learning*, 18.

Jeong, K. S., S. H. Myaeng, J. S. Lee, and K. S. Choi, 1999. Automatic identification and back-transliteration of foreign words for information retrieval. *Information Processing and Management*, 35(4):523-540.

Kang, Y. and A. A. Maciejewski, 1996. An Algorithm for Generating a Dictionary of Japanese Scientific Terms. *Literary and Linguistic Computing*, 11(2).

Knight, K. and J. Graehl, 1997. Machine Transliteration. In *Proceedings. of the 35th Annual Meetings of the Association for Computational Linguistics (ACL)*, Madrid, Spain.

Lee, J. S., 1999. An English-Korean Transliteration and Retransliteration Model for Cross-lingual Information Retrieval. Ph.D. dissertation, Dept. of Computer Science, Korea Advanced Institute of Science and Technology.

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

Mingers, J., 1989. An empirical comparison of selection measures for decision tree induction. *Machine Learning*, 3:319-342.

Mingers, J., 1989. An empirical comparison of pruning methods for decision tree induction. *Machine Learning*, 4:227-243.

Mitchell, T. M., 1997. *Machine Learning*. The McGraw-Hill Companies, Inc.

Nam, Y. S., 1997. *The latest foreign word dictionary*. Sung-An-Dang Press.

Quinlan, J. R., 1987. Rule induction with statistical data – a comparison with multiple regression. *Journal of the Operational Research Society*, 38:347-352.

Quinlan, J. R., 1986. Induction of decision trees. *Machine Learning*, 1:81-106.

Stalls, B. and K. Knight, 1998. Translating Names and Technical Terms in Arabic Text. In *Proceedings of the COLING/ACL Workshop on Computational Approaches to Semitic Languages.*

Wagner, R. A. and M. J. Fischer, 1974. The string-to-string correction problem. *Journal of ACM* 21(1):168-178.

Wan, S. and C. M. Verspoor, 1998. Automatic English-Chinese name transliteration for development of multilingual resources. In *Proceedings of COLING-ACL'98, the joint meeting of 17th International Conference on Computational Linguistics and the 36th Annual Meeting of the Association for Computational Linguistics*, Montreal, Canada.