# Machine Transliteration Using Multiple Transliteration Engines and Hypothesis Re-Ranking

**Jong-Hoon Oh and Hitoshi Isahara**

Computational Linguistics Group
National Institute of Information and Communications Technology
3-5 Hikaridai Seika-cho, Soraku-gun, Kyoto 619-0289 Japan
{rovellia, isahara}@nict.go.jp

**Abstract**

This paper describes a novel method of improving machine transliteration by using multiple transliteration hypotheses and re-ranking them. We constructed seven machine-transliteration engines to produce a set of transliteration hypotheses. We then re-ranked the hypotheses to select the correct transliteration hypothesis. We propose a re-ranking method that makes use of confidence-score, language-model, and Web-frequency features and combines them with machine-learning algorithms including support vector machines and the maximum entropy model. Our testing of English-to-Japanese and English-to-Korean transliterations revealed that the individual transliteration engines used in our approach performed comparably to previous approaches and that re-ranking improved word accuracy compared to the best individual engine from about 65 to 88%.

## 1. Introduction

Transliteration is particularly used to translate proper names and technical terms from languages using the Roman alphabet into ones using non-Roman alphabets such as Chinese, Japanese, or Korean. Because transliteration is one of the main causes of the out-of-vocabulary (OOV) problem, machine transliteration has received a significant degree of attention as a tool to support machine translation (Knight and Graehl, 1998; Al-Onaizan and Knight, 2002) and cross-language information retrieval (Fujii and Tetsuya, 2001). A variety of paradigms for machine transliteration have been developed over the years: *grapheme*[1]*-based model* (GM) (Kang and Kim, 2000; Goto et al., 2003), *phoneme*[2]*-based model* (PM) (Knight and Graehl, 1998; Kang, 2001), *hybrid model* (HM) (Al-Onaizan and Knight, 2002; Bilac and Tanaka, 2004), and *correspondence-based model* (CM) (Oh and Choi, 2002; Oh and Choi, 2005). These models are classified in terms of the information sources used for transliteration or the units that are transliterated. GM, PM, HM, and CM make use of source graphemes, source phonemes, both source graphemes and source phonemes, and the correspondence between source graphemes and phonemes, respectively.

Transliteration is generally a phonetic rather than an orthographic process (Knight and Graehl, 1998). However, both the source grapheme and source phoneme or either of them can affect the target-language transliteration (e.g. a Japanese or Korean transliteration). For this reason, there are transliterations that are grapheme-based, ones that are phoneme-based, and ones that are a combination of grapheme-based and phoneme-based transliterations. For example, the respective Korean transliterations of *data*, *amylase*, and *neomycin* are the phoneme-based transliteration 'de-i-teo (데이터)'[3], the grapheme-based transliteration 'a-mil-la-a-je (아밀라아제)', and 'ne-o-ma-i-sin (네오마이신)', which is a combination of the grapheme-based transliteration 'ne-o (네오)' and the phoneme-based transliteration 'ma-i-sin (마이신)'. However, because each of the transliteration models depends on a particular information source, each can produce transliterations with errors. Moreover, different transliteration models usually produce different errors and different transliterations. This means we should be able to improve transliteration by combining various transliteration models into one machine-transliteration system that combines the advantages of the individual models and suffers from few of their disadvantages.

A similar idea has been successfully applied to automatic speech recognition (ASR) and machine translation (Fiscus, 1997; Nomoto, 2004). In our previous work (Oh et al., 2006b), we have shown that this idea is also helpful for improving machine transliteration. It used transliteration hypotheses derived from four transliteration engines and re-ranked them with the product of two ranking functions, each of which was based on the rank of hypotheses in each transliteration engine and Web frequency. Even though the re-ranking in Oh et al. (2006b) performed well, it had limitations in taking various features into account and effectively combining them. To address this problem, we developed SVM-based and MEM-based re-ranking methods, which are able to effectively combine various features.

We describe our framework in Sections 2. and 3. We then describe our evaluation in Section 4. and review related work in Section 5. The paper is concluded in Section 6.

## 2. Producing Transliteration Hypotheses

We used multiple transliteration engines based on GM, PM, HM, and CM to produce transliteration hypotheses. GM, PM, and CM can generally function alone as transliteration engines, while HM depends on other transliteration models to estimate its parameters. Therefore, we called a

[1] Graphemes refer to the basic units (or the smallest contrastive units) of a written language: e.g., English has 26 graphemes or letters

[2] Phonemes are the simplest significant unit of sound.

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

transliteration engine based on GM, PM, or CM a "single-model engine" and one based on HM a "hybrid-model engine." We used seven transliteration engines. Three were *single-model engines* corresponding to GM, PM, and CM. The other four were *hybrid-model engines*. Three of these corresponded to HM using two of GM, PM, and CM — $\text{HM}_{(\mathcal{G}+\mathcal{P})}$, $\text{HM}_{(\mathcal{G}+\mathcal{C})}$, and $\text{HM}_{(\mathcal{P}+\mathcal{C})}$ — and the last was based on HM using all three ($\text{HM}_{(\mathcal{G}+\mathcal{P}+\mathcal{C})}$). Note that $\text{HM}_{(\mathcal{G}+\mathcal{P})}$ has previously been described (Al-Onaizan and Knight, 2002; Bilac and Tanaka, 2004), and the other HMs are newly proposed here.

### 2.1. Single-Model Engines

Let $SW$ be a source word, $P_{SW}$ be the pronunciation of $SW$, $T_{SW}$ be a target word corresponding to $SW$, and $C_{SW}$ be the correspondence between $SW$ and $P_{SW}$. $P_{SW}$ and $T_{SW}$ can be segmented into a series of sub-strings, each of which corresponds to a source grapheme. We can thus write $SW = s_1, \cdots, s_n = s_1^n$, $P_{SW} = p_1, \cdots, p_n = p_1^n$, $T_{SW} = t_1, \cdots, t_n = t_1^n$, and $C_{SW} = c_1, \cdots, c_n = c_1^n$, where $s_i$, $p_i$, $t_i$, and $c_i = < s_i, p_i >$ respectively represent the $i^{th}$ source grapheme, source phonemes corresponding to $s_i$, target graphemes corresponding to $s_i$ and $p_i$, and the correspondence between $s_i$ and $p_i$. Table 1 shows an example of correspondence between $s_i$, $p_i$, and $t_i$, where $SW = acetylcholine$, $P_{SW}$="AH S EH T AH L K OW L IY N"[4], and $T_{SW}$= 'a-se-ti-ru-ko-rin (アセチルコリン)' in Japanese, and $T_{SW}$='a-se-til-kol-lin (아세틸콜린)' in Korean. With this definition, GM ($SW \rightarrow T_{SW}$), PM ($SW \rightarrow P_{SW}$ and $P_{SW} \rightarrow T_{SW}$), and CM ($SW \rightarrow P_{SW}$ and $C_{SW} \rightarrow T_{SW}$) can respectively be represented as Eqs. (1), (2), and (3). Given the assumption that each transliteration model depends on the size of the context, $k$, Eqs. (1), (2), and (3) can be simplified into a series of products.

$$
\begin{aligned}
Pr_{\mathcal{G}} &= Pr_{\mathcal{G}}(T_{SW}|SW) = Pr(t_1^n|s_1^n) \qquad (1)\\
&\approx \prod_i Pr(t_i|t_{i-k}^{i-1}, s_{i-k}^{i+k})\\
Pr_{\mathcal{P}} &= Pr_{\mathcal{P}}(T_{SW}|SW) \qquad (2)\\
&= Pr(p_1^n|s_1^n) \times Pr(t_1^n|p_1^n)\\
&\approx \prod_i Pr(p_i|p_{i-k}^{i-1}, s_{i-k}^{i+k}) \times Pr(t_i|t_{i-k}^{i-1}, p_{i-k}^{i+k})\\
Pr_{\mathcal{C}} &= Pr_{\mathcal{C}}(T_{SW}|SW) \qquad (3)\\
&= Pr(p_1^n|s_1^n) \times Pr(t_1^n|c_1^n)\\
&\approx \prod_i Pr(p_i|p_{i-k}^{i-1}, s_{i-k}^{i+k}) \times Pr(t_i|t_{i-k}^{i-1}, c_{i-k}^{i+k})
\end{aligned}
$$

To estimate the probabilities, $Pr(t_i|t_{i-k}^{i-1}, s_{i-k}^{i+k})$, $Pr(p_i|p_{i-k}^{i-1}, s_{i-k}^{i+k})$, $Pr(t_i|t_{i-k}^{i-1}, p_{i-k}^{i+k})$, and $Pr(t_i|t_{i-k}^{i-1}, c_{i-k}^{i+k})$, in Eqs. (1), (2), and (3), we use the maximum entropy model (Berger et al., 1996). Event $ev$ in this model is composed of a target event ($te$) and a history event ($he$) and is represented by a bundle of feature functions ($f_i(he, te)$) that represent certain characteristics in event $ev$. The feature functions enable a model based on the maximum entropy model to estimate probability (Berger et al., 1996). Therefore, designing the feature functions, which effectively support certain decisions made by the model, is important. Our basic strategy in designing the feature functions was that the context information collocated with the unit of interest was important. On the basis of this strategy, we designed feature functions with the following features. Table 2 shows examples of feature functions based on the following features.

- Single features in $s_{i-k}^{i+k}$, $p_{i-k}^{i+k}$, $c_{i-k}^{i+k}$, and $t_{i-k}^{i-1}$ (e.g., $s_i$, $p_i$, $c_i$, and $t_{i-1}$)
- Combinations between features of the same type including bigram and trigram (e.g., $\{s_{i-2}^{i}\}$, $\{p_{i-2}, p_i, p_{i+2}\}$, $\{c_i^{i+2}\}$, and $\{t_{i-2}^{i-1}\}$)
- Combinations between features of different type (e.g., $\{s_{i-1}^{i}, p_i\}$, $\{s_{i-1}, t_{i-2}^{i-1}\}$, $\{c_i, t_{i-1}\}$, and $\{p_{i-3}^{i-2}, t_{i-2}\}$)
  - between $s_{i-k}^{i+k}$ and $p_{i-k}^{i+k}$
  - between $s_{i-k}^{i+k}$ and $t_{i-k}^{i-1}$
  - between $p_{i-k}^{i+k}$ and $t_{i-k}^{i-1}$
  - between $c_{i-k}^{i+k}$ and $t_{i-k}^{i-1}$

A conditional maximum entropy model is generally an exponential log-linear model that gives the conditional probability of event $ev =< te, he >$, as described in Eq. (4), where $\lambda_i$ is the parameter to be estimated (Berger et al., 1996).

$$Pr(te|he) = \frac{exp(\sum_i \lambda_i f_i(he, te))}{\sum_{te} exp(\sum_i \lambda_i f_i(he, te))} \qquad (4)$$

Using Eq. (4) and feature functions, we can estimate the conditional probabilities in Eqs. (1), (2), and (3), as $Pr(t_i|t_{i-k}^{i-1}, c_{i-k}^{i+k}) = Pr(te|he)$, because we can respectively represent $te$ and $he$ as $t_i$ and tuples $< t_{i-k}^{i-1}, c_{i-k}^{i+k} >$. In the same way, $Pr(t_i|t_{i-k}^{i-1}, s_{i-k}^{i+k})$, $Pr(t_i|t_{i-k}^{i-1}, p_{i-k}^{i+k})$, and $Pr(p_i|p_{i-k}^{i-1}, s_{i-k}^{i+k})$ can be represented as $Pr(te|he)$ with their corresponding target events and history events. We used the "Maximum Entropy Modeling Toolkit"[5] to estimate the probabilities and the LBFGS algorithm to find $\lambda_i$ in Eq. (4).

### 2.2. Hybrid-Model Engines

Using the definition of HM in Al-Onaizan and Knight (2002) and Bilac and Tanaka (2004), we can represent four hybrid-model engines in a straightforward manner — Eqs. (5), (6), (7), and (8), where $0 < \alpha, \beta, \gamma, \delta_1, \delta_2, \delta_3 < 1$ and $\delta_1 + \delta_2 + \delta_3 = 1$. Note that $Pr_{\mathcal{G}}$, $Pr_{\mathcal{P}}$, and $Pr_{\mathcal{C}}$ in Eqs. (5), (6), (7), and (8) correspond to Eqs. (1), (2), and (3), respectively.

[4] *ARPAbet* symbols are used to represent English phonemes. *ARPAbet* is one of the methods used for coding English phonemes into ASCII characters (`http://www.cs.cmu.edu/~laura/pages/arpabet.ps`)

[5] Available at `http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html`

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| English | $c_i$ | $s_i$ | $a$ | $c$ | $e$ | $t$ | $y$ | $l$ | $c$ | $h$ | $o$ | $l$ | | $i$ | $n$ | $e$ |
| | | $p_i$ | AH | S | EH | T | AH | L | K | $\varepsilon$ | OW | L | | IY | N | $\varepsilon$ |
| Japanese | $t_i$ | | 'a' | 's' | 'e' | 't' | 'i' | 'ru' | 'k' | $\varepsilon$ | 'o' | 'r' | | 'i' | 'n' | $\varepsilon$ |
| | $ts_j$ | | ア | セ | | チ | | ル | コ | | | リ | | ン | | |
| Korean | $t_i$ | | 'a' | 's' | 'e' | 't' | 'i' | 'l' | 'k' | $\varepsilon$ | 'o' | 'l' | 'l' | 'i' | 'n' | $\varepsilon$ |
| | $ts_j$ | | 아 | 세 | | 틸 | | | 콜 | | | 린 | | | | |

Table 1: Examples of correspondence between $s_i$, $p_i$, and $t_i$. $\varepsilon$ means a NULL character and $ts_j$ represents target-language syllables.

| $f_j$ | $s_{i-k}^{i+k}$ | $p_{i-k}^{i+k}$ | $t_{i-k}^{i-1}$ | $t_i$ |
|---|---|---|---|---|
| $f_1$ | $s_{i-1}^{i+1} = cet$ | – | $t_{i-1}$='s' | $t_i$='e' |
| $f_2$ | $c_{i-1}^{i} =<ce,$"S EH"> | | – | $t_i$='e' |
| $f_3$ | – | $p_i$="EH" | – | $t_i$='e' |
| $f_4$ | $s_i^{i+1} = et$ | $p_i^{i+2}$="EH T AH" | – | $t_i$='e' |
| $f_5$ | $s_{i-2}^{i-1} = ac$ and $s_{i+1} = t$ | – | $t_{i-2}^{i-1}$='a-s' | $t_i$='e' |
| $f_6$ | $c_i =< e,$"EH"> | | $t_{i-2}^{i-1}$='a-s' | $t_i$='e' |
| $f_7$ | $s_i = e$ | $p_{i-3}$=$ and $p_i^{i+1}$="EH T" | – | $t_i$='e' |

Table 2: Examples of feature functions derived from Table 1. $ represents the start of words

$$Pr_{\mathcal{HM}(\mathcal{G}+\mathcal{P})}(T_{SW}|SW) = \alpha \times Pr_{\mathcal{P}} + (1-\alpha) \times Pr_{\mathcal{G}} \quad (5)$$

$$Pr_{\mathcal{HM}(\mathcal{G}+\mathcal{C})}(T_{SW}|SW) = \beta \times Pr_{\mathcal{C}} + (1-\beta) \times Pr_{\mathcal{G}} \quad (6)$$

$$Pr_{\mathcal{HM}(\mathcal{P}+\mathcal{C})}(T_{SW}|SW) = \gamma \times Pr_{\mathcal{C}} + (1-\gamma) \times Pr_{\mathcal{P}} \quad (7)$$

$$Pr_{\mathcal{HM}(\mathcal{G}+\mathcal{P}+\mathcal{C})}(T_{SW}|SW) = \delta_1 \times Pr_{\mathcal{G}} + \delta_2 \times Pr_{\mathcal{P}} + \delta_3 \times Pr_{\mathcal{C}} \quad (8)$$

We first produce $n$-best transliteration hypotheses using a stack decoder (Schwartz and Chow, 1990) for each transliteration engine. We then make a set of transliteration hypotheses comprising the $n$-best transliteration hypotheses produced by the seven transliteration engines.

## 3. Re-Ranking Transliteration Hypotheses

The transliteration hypotheses in the set are re-ranked to enable a correct hypothesis to be identified. Re-ranking has been successfully applied to several NLP problems including statistical parsing (Collins and Koo, 2005; Daume III and Marcu, 2004; Shen and Joshi, 2003), machine translation (Shen et al., 2004), and name entity taggers (Ji et al., 2006). We selected support vector machines (SVMs) and the maximum entropy model (MEM) (Daume III and Marcu, 2004; Ji et al., 2006) from the machine-learning algorithms used for re-ranking.

Let $h_i \in H$ be the $i^{th}$ transliteration hypothesis of source word $s$, $h_{correct}$ be a correct transliteration hypothesis corresponding to $s$, $x_i \in X$ be a feature vector of $h_i$, and $y_i$ be the training label for $x_i$. What we need to do is devise a rank function, $g(x_i)$, in Eq. (9) that ranks $h_{correct}$ higher and the others lower.

$$g(x_i) : X \rightarrow \{r : r\ is\ ordering\ of\ h_i \in H\} \quad (9)$$

We first train SVMs and MEM with training data set $D = \{x_i, y_i\}$, where $x_{correct}$ is a positive sample ($y_{correct} = positive$) and $x_{i \neq correct}$ is a negative sample ($y_{i \neq correct} = negative$). The SVMs assign a value to each transliteration hypothesis ($h_i$) using

$$g_{SVM}(x_i) = w \cdot x_i + b \quad (10)$$

where $w$ denotes a weight vector. We did not use the predicted class of $x_i$ in SVM-based re-ranking but the predicted value of $g_{SVM}(x_i)$ because our re-ranking function, as represented by Eq. (9), determines the relative ordering between $h_i$ and $h_j$ in $H$. A re-ranking function based on MEM assigns probability to $h_i$ using

$$g_{MEM}(x_i) = Pr(h_{correct}|x_i) \quad (11)$$

We can finally obtain a ranked list for given $H$ and $X$ — the higher the $g(x_i)$ value, the better the $h_i$. We used $SVM^{light}$ (Joachims, 2002) and the "Maximum Entropy Modeling Toolkit" for our re-ranking.

### 3.1. Features for Re-ranking

We needed to design a suitable feature to measure the relevance between a source word, $s$, and a transliteration hypothesis, $h_i$, of $s$ to train the SVMs and MEM for our re-ranking. We introduced three types of features.

**Confidence Score (8 features toal)**: Each transliteration engine produces transliteration hypotheses and their corresponding confidence scores using Eqs. (1)–(8). We use the scores as a feature for re-ranking. Let $CS_{\mathcal{M}}(s, h_i)$ be a confidence score function in $\mathcal{M} \in \{\mathcal{G}, \mathcal{P}, \mathcal{C}, \mathcal{HM}(\mathcal{G}+\mathcal{P}), \mathcal{HM}(\mathcal{G}+\mathcal{C}), \mathcal{HM}(\mathcal{P}+\mathcal{C}),$ and $\mathcal{HM}(\mathcal{G}+\mathcal{P}+\mathcal{C})\}$, each of which corresponds to Eqs. (1), (2), (3), (5), (6), (7), and (8), and $ACS_{\mathcal{M}}(s, h_i)$ be the average of $CS_{\mathcal{M}}(s, h_i)$ over $\mathcal{M}$. The confidence score features for $h_i$ can then be ac-

quired using Eq. (12).

$$CS_{\mathcal{M}}(s,h_i) = Pr_{\mathcal{M}}(h_i|s) \quad (12)$$
$$ACS(s,h_i) = \frac{1}{|\mathcal{M}|} \times \sum_{\mathcal{M}} CS_{\mathcal{M}}(s,h_i)$$

**Language Model (1 feature total)**: We used a list of transliterations as mono-lingual corpora to construct the language model for transliteration. Note that the mono-lingual corpora differed from the training data set used to train the transliteration engines. We used a list of Korean transliterations published by "The National Institute of the Korean Language"[6] for the Korean language model and a list of katakana terms extracted from Web texts used in Kawahara and Kurohashi (2006) for the Japanese language model. Note that katakana is usually used to represent Japanese transliterations. We then construct a transliteration language model using the SRI Language Modeling Toolkit (SRILM) (Stolcke, 2002). Let $h_i = ts_1, \cdots, ts_l$ be a transliteration hypothesis having $l$ target language syllables. We calculate $Pr(h_i) = logPr(ts_1, \cdots, ts_l)$ using SRILM. Note that the language-model features are based on target-language syllables rather than target-language graphemes.

**Web Frequency (6 features total)**: Several researchers have used Web frequency, i.e., the number of Web documents retrieved by a search engine, for transliteration or ranking translations (Al-Onaizan and Knight, 2002; Qu and Grefenstette, 2004; Zhang et al., 2005; Oh et al., 2006b). There have been several methods of acquiring Web frequency. For source word $s$ and target transliteration (or translation) $t$, some of them have used $t$ as a query for Web frequency (Al-Onaizan and Knight, 2002; Oh et al., 2006b), which is called a *monolingual keyword search* (MKS), and others have used both $s$ and $t$ for acquiring Web frequency (Qu and Grefenstette, 2004; Zhang et al., 2005; Oh et al., 2006b), which is called a *bilingual keyword search* (BKS). However, Web pages retrieved by MKS tend to show whether $t$ is used in target-language texts rather than whether $t$ is a translation or transliteration of $s$. BKS frequently retrieves Web pages where $s$ and $t$ have little relation to each other because it does not consider the distance between them. To address this problem, Oh et al. (2006b) used a phrase composed of $s$ and $t$ as a query, called a *bilingual phrasal search* (BPS). $s$ and $t$ tend to be close together in the texts of Japanese and Korean Web pages if they are counterparts, such as "si-na-pu-su (シナプス) (*synapse*)" and "si-naep-seu (시냅스) (*synapse*)." Therefore, the constraint in BPS — $s$ and $t$ must form a phrase in the retrieved Web pages — is very helpful for transliteration re-ranking (Oh et al., 2006b).

Let $WF_{\mathcal{WS}}(s,t)$ be the Web frequency in $\mathcal{WS} = \{MKS, BKS, BPS\}$, and $h_i \in H$ be the $i^{th}$ transliteration hypothesis of source word $s$. Relative Web frequency is defined as

$$RWF_{\mathcal{WS}}(s,h_i) = \frac{WF_{\mathcal{WS}}(s,h_i)}{\sum_{h_k \in H} WF_{\mathcal{WS}}(s,h_k)} \quad (13)$$

[6] "The usage of Korean transliterations" at `http://www.korean.go.kr`

We used six Web frequency features, $WF_{\mathcal{WS}}(s,h_i)$, and $RWF_{\mathcal{WS}}(s,h_i)$.

# 4. Evaluation

We evaluated the effectiveness of our system for English-to-Japanese and English-to-Korean transliterations. The test data for the English-to-Japanese transliteration (EJSet), which consisted of English-katakana pairs from EDICT[7], consisted of 10,417 pairs. The test data for the English-to-Korean transliteration (EKSet) (Nam, 1997) consisted of 7,172 English-Korean pairs. EJSet and EKSet covered proper names, technical terms, and general terms.

## 4.1. Experimental Procedure

| | EKSet | EJSet |
|---|---|---|
| Training Set | 5,124 | 8,335 |
| Development Set | 1,024 | 1,041 |
| Blind Test Set | 1,024 | 1,041 |

Table 3: Test data sets

We divided the EJSet and EKSet into $k$ subsets of equal size and constructed a training set with $k-2$ subsets, a development set with one subset, and a blind test set with the remaining subset (see Table 3), where $k = 10$ for EJSet and $k = 7$ for EKSet. The training set was used to train the seven transliteration engines. The development set was used to determine the number of hypotheses of each transliteration engine and the linear interpolation parameters of the hybrid-model engines at ($\alpha$, $\beta$, $\gamma$, $\delta_1$, $\delta_2$, and $\delta_3$)[8], and to train the SVMs and MEM used for re-ranking. We used the blind test set for evaluation.

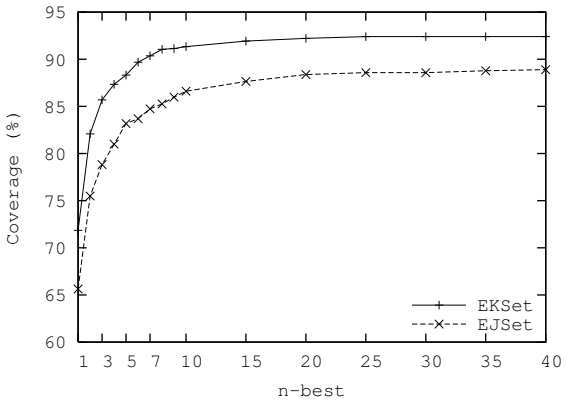


Figure 1: Coverage on basis of $n$-best

We tested *coverage*, which is the proportion of correct transliterations in the transliteration hypotheses of the seven

[7] `http://www.csse.monash.edu.au/~jwb/edict.html`

[8] We determine the parameters having maximum word accuracy or minimum error rate. $\alpha = 0.3$, $\beta = 0.3$, $\gamma = 0.8$, $\delta_1 = 0.5$, $\delta_2 = 0.2$, and $\delta_3 = 0.3$ for EKSet and $\alpha = 0.4$, $\beta = 0.7$, $\gamma = 0.7$, $\delta_1 = 0.3$, $\delta_2 = 0.1$, and $\delta_3 = 0.6$ for EJSet.

transliteration engines to the transliterations in the development set to determine $n$-best (the number of hypotheses for each transliteration engine). Although the results in Fig. 1 indicate that the bigger the $n$-best, the higher the coverage, the coverage converges at $n$-best=10. Therefore, we used $n$-best=10 in our experiments.
The evaluation was done in terms of *word accuracy* ($WA$). $WA$ is the proportion of correct transliterations in the best hypothesis to correct transliterations in the blind test set.

### 4.2. Results

| System | | EJSet | EKSet |
|---|---|---|---|
| Previous Work | KANG00 | 52.2 | 54.1 |
| | GOTO03 | 54.3 | 55.3 |
| | KANG01 | 50.7 | 47.5 |
| | BILAC04 | 57.1 | 59.3 |
| Individual Transliteration Engines | GM($\mathcal{G}$) | 61.6 | 59.0 |
| | PM($\mathcal{P}$) | 54.4 | 56.7 |
| | CM($\mathcal{C}$) | 65.0 | 65.1 |
| | HM$_{(\mathcal{G}+\mathcal{P})}$ | 62.9 | 61.3 |
| | HM$_{(\mathcal{G}+\mathcal{C})}$ | 64.5 | 62.6 |
| | HM$_{(\mathcal{P}+\mathcal{C})}$ | 64.6 | 64.7 |
| | HM$_{(\mathcal{G}+\mathcal{P}+\mathcal{C})}$ | 62.0 | 62.4 |

Table 4: Comparison of individual transliteration engines (%)

We compared the performance of our individual transliteration engines against that of four reported ones, shown in the top section of Table 4. KANG00 and GOTO03 are the transliteration-network based systems of Kang and Kim (2000) and Goto et al. (2003), which they modeled using GM. KANG01 is the decision-tree based system of Kang (2001), which he modeled using PM. Bilac and Tanaka (2004) used HM$_{(\mathcal{G}+\mathcal{P})}$ for BILAC04. We trained and tested these systems using the same test data as for our individual transliteration engines. The bottom of the table shows the $WA$ for all transliteration engines used in our system, which are the results for our system without re-ranking.
These results show that if the previous systems and the individual transliteration engines are based on the same transliteration model, they had comparable performance or our individual transliteration engines are better than the previous systems — KANG00 and GOTO03 vs. GM, KANG01 vs. PM, and BILAC04 vs. HM$_{(\mathcal{G}+\mathcal{P})}$. Looking at the performances of the individual transliteration engines, CM showed the best performance and GM and PM showed little bit lower performance than CM and HM. As described by Oh et al. (2006a), machine transliteration engines using both source graphemes and phonemes are more accurate than those based on either source graphemes or phonemes.
The results with re-ranking are listed in Table 5. We compared how well our method performed against those of the four previous systems — the first was ROVER (Fiscus, 1997) and the other three were based on RWF (Oh et al., 2006b; Oh et al., 2006a). ROVER is the most popular method used in automatic speech recognition (ASR) system combinations. It is based on simple voting schemes

| System | | EJSet | EKSet |
|---|---|---|---|
| Single-best | | 65.0 | 65.1 |
| Previous work | ROVER | 66.8 | 66.4 |
| | $RWF_{\mathcal{MKS}}$ | 45.7 | 37.5 |
| | $RWF_{\mathcal{BKS}}$ | 77.9 | 76.5 |
| | $RWF_{\mathcal{BPS}}$ | 85.4 | 85.0 |
| Our approach | MEM | 87.4 | 87.5 |
| | SVM | 87.8 | 88.2 |
| Upper Bound | | 93.6 | 93.6 |

Table 5: Comparison of hypothesis re-reanking (%)

over only the top hypothesis from each system — majority voting (if most systems agree on a certain word in a particular position, the word is selected) and confidence voting (voting by the confidence scores produced by ASR systems). We used a ROVER system implemented in the "NIST Scoring ToolKit (SCTK)"[9] to obtain the results in Table 5. The three RWF systems in Table 5 are based on Eq. (13), which Oh et al. (2006b) used to re-rank transliterations. The single-best system represents an individual transliteration engine having the best performance (CM in Table 4). MEM and SVM in our approach represent the performance of re-ranking based on MEM and SVM. The upper bound is a system that always outputs a correct hypothesis as the best if there is a correct transliteration in the set of hypotheses produced by individual transliteration engines. Therefore, the upper-bound results are the upper bounds for how well our system performed with re-ranking.

Both RWF and ROVER outperformed the single-best system except for $RWF_{\mathcal{MKS}}$. As Oh et al. (2006b) reported, $RWF_{\mathcal{MKS}}$ could not effectively re-rank transliteration hypotheses because it could not take the source-language terms of the hypotheses in retrieving Web pages into account. We investigated what effect Web-search methods would have on the re-ranking hypotheses in Section 4.2.2. Our re-ranking systems outperformed RWF and ROVER. These improvements might have occurred because our re-ranking systems used more features than ROVER and RWF, and effectively combined these features. ROVER used a confidence value similar to our confidence-score feature and RWF used one of our Web-frequency features. Our re-ranking systems took three features including confidence-score, language-model, and Web-frequency features into consideration. Our system also effectively combined the three features using SVM and MEM and thus performed the best. The details on performance depending on features is discussed in Section 4.2.1.

MEM and SVM performed significantly better than the individual transliteration engines. Our approach effectively re-ranked the transliteration hypotheses so that it performed much closer to the upper bound than any of the individual transliteration engines.

| | EJSet | | EKSet | |
|---|---|---|---|---|
| | MEM | SVM | MEM | SVM |
| CS | 68.2 | 67.9 | 66.5 | 67.0 |
| LM | 27.5 | 22.4 | 34.1 | 37.2 |
| WF | 78.0 | 85.0 | 82.1 | 85.7 |
| CS+LM | 70.3 | 72.0 | 70.4 | 71.5 |
| CS+WF | 86.9 | 87.0 | 87.0 | 87.0 |
| LM+WF | 83.8 | 85.1 | 86.8 | 87.3 |
| CS+LM+WF | 87.4 | 87.8 | 87.5 | 88.2 |
| Upper Bound | 93.6 | | 93.6 | |

Table 6: Contribution of features in re-ranking (%)

| | EJSet | | EKSet | |
|---|---|---|---|---|
| | MEM | SVM | MEM | SVM |
| Baseline | 70.3 | 72.0 | 70.4 | 71.5 |
| MKS | 77.4 | 76.8 | 72.6 | 73.2 |
| BKS | 84.3 | 83.2 | 81.3 | 82.9 |
| BPS | 86.2 | 86.6 | 86.4 | 87.2 |
| MKS+BKS | 83.6 | 83.5 | 81.5 | 83.0 |
| MKS+BPS | 86.4 | 86.8 | 86.6 | 87.1 |
| BKS+BPS | 87.6 | 87.2 | 87.3 | 87.7 |
| MKS+BKS+BPS | 87.4 | 87.8 | 87.5 | 88.2 |
| Upper Bound | 93.6 | | 93.6 | |

Table 7: Effect of Web-search methods (%)

#### 4.2.1. Contributions of Features in Re-ranking

We experimented with different feature settings for re-ranking to investigate what contribution they made to re-ranking. Table 6 lists the results. CS, LM, and WF are systems that use confidence-score, language-model, and Web-frequency features. CS+LM, CS+WF, and LM+WF use two of the confidence-score, language-model, and Web-frequency features. CS+LM+WF is equivalent to "Our approach" in Table 5.

WF outperformed CS, LM, and WF but LM performed the worst. However, combining the features consistently improved performance — CS+LM+WF was the best and CS+LM, CS+WF, and LM+WF performed better than CS, LM, and WF. This means that the greater the number of features, the higher the performance. We could estimate the contribution made by each feature by calculating the gap in performance between CS+LM+WF and a system with a combination of two features. We found that CS+LM+WF and CS+LM had the largest gap in performance, thus WF might have contributed the most to re-ranking. However, CS and LM also made a positive contribution to CS+LM+WF, even though their contribution was lower than that of WF. WF can be viewed as a language or translation model on the word level, while LM is a language model on the syllable level. The roles of WF and LM in re-ranking hypotheses overlapped to a certain degree. Moreover, WF outperformed LM in re-ranking hypotheses. For these reasons, LM made less contribution than WF.

#### 4.2.2. Effect of MKS, BKS, and BPS

We experimented with different settings for the Web-frequency features in our re-ranking function to clarify what effect MKS, BKS, and BPS had. Table 7 lists the results for a baseline system, our system with seven different Web-frequency features, and the upper-bound system. The baseline system corresponds to CS+LM in Table 6. The upper bound is the same system as that in Tables 5 and 6. The seven versions of our system in the tables use different Web-frequency features but share the same features used in the baseline system. MKS, BKS, and BPS are systems that use Web-frequency features acquired using one of MKS, BKS, or BPS. MKS+BKS, MKS+BPS, and BKS+BPS use Web-frequency features acquired using two of MKS, BKS, and BPS. MKS+BKS+BPS is equivalent to the CS+LM+WF in Table 6.

The systems based on different Web-frequency features yielded different results (Table 7). MKS was the worst, and BPS was the best of MKS, BKS, and BPS. Moreover, it is evident that the systems that used Web-frequency features acquired by BPS — MKS+BPS, BKS+BPS, and MKS+BKS+BPS — tended to perform well. This means that BPS is better able to retrieve reliable Web pages for re-ranking transliteration hypotheses than BKS and MKS. However, $\sum_{h_k \in H} WF_{\mathcal{WS}}(s, h_k) = 0$ occurs in BPS more often than in BKS and MKS. It can reduce the effectiveness of re-ranking the transliteration hypotheses when BPS is used alone because $WF_{BPS}$ and $RWF_{BPS}$ become zero for all hypotheses in $H$. However, we can overcome this problem by using Web-frequency features acquired by BPS along with those acquired by BKS and MKS. Thus, we can obtain the best results with MKS+BKS+BPS.

#### 4.2.3. Effect of Number of Transliteration Engines

| $\|R\|$ | EJSet | | | EKSet | | |
|---|---|---|---|---|---|---|
| | MEM | SVM | UB | MEM | SVM | UB |
| 1 | 80.6 | 80.8 | 85.9 | 81.7 | 81.8 | 86.2 |
| 2 | 85.8 | 86.3 | 92.0 | 86.3 | 86.5 | 92.3 |
| 3 | 87.7 | 87.7 | 93.5 | 87.4 | 87.5 | 93.5 |
| 4 | 87.9 | 87.7 | 93.6 | 87.4 | 87.3 | 93.5 |
| 5 | 87.9 | 87.7 | 93.6 | 87.8 | 87.6 | 93.6 |
| 6 | 87.8 | 87.5 | 93.6 | 87.7 | 87.6 | 93.6 |
| 7 | 87.7 | 87.5 | 93.6 | 87.5 | 88.2 | 93.6 |

Table 8: Effect of number of transliteration engines (%)

We investigated what effect the number of transliteration engines ($|R|$) would have on transliteration by increasing $|R|$ from 1 to 7, starting from $GM$ and adding $PM$, $CM$, $HM_{(\mathcal{G}+\mathcal{P})}$, $HM_{(\mathcal{G}+\mathcal{C})}$, $HM_{(\mathcal{P}+\mathcal{C})}$, and $HM_{(\mathcal{G}+\mathcal{P}+\mathcal{C})}$ in that order. The UB in Table 8 represents the results for upper-bound systems like those in Tables 5, 6, and 7. Figures 2 and 3 plot how well upper-bound systems perform according to $n$-best and the number of transliteration engines.

Multiple transliteration engines usually produce more transliteration hypotheses than individual ones; we would

[9] available at http://www.nist.gov/speech/tools/index.htm

| | GM($\mathcal{G}$) | | PM($\mathcal{P}$) | | CM($\mathcal{C}$) | | HM$_{(\mathcal{G}+\mathcal{P})}$ | | HM$_{(\mathcal{G}+\mathcal{C})}$ | | HM$_{(\mathcal{P}+\mathcal{C})}$ | | HM$_{(\mathcal{G}+\mathcal{P}+\mathcal{C})}$ | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EJ | EK | EJ | EK | EJ | EK | EJ | EK | EJ | EK | EJ | EK | EJ | EK |
| GM($\mathcal{G}$) | *85.9* | *86.2* | 92.0 | 91.9 | 92.3 | 92.3 | 91.3 | 89.2 | 92.1 | 89.2 | 93.0 | 92.3 | 92.7 | 90.2 |
| PM($\mathcal{P}$) | | | *85.8* | *83.0* | 90.5 | 90.1 | 90.6 | 91.5 | 91.2 | 92.1 | 90.2 | 89.2 | 91.0 | 91.8 |
| CM($\mathcal{C}$) | | | | | *88.6* | *89.0* | 92.9 | 92.4 | 90.4 | 92.2 | 90.2 | 89.4 | 91.2 | 92.4 |
| HM$_{(\mathcal{G}+\mathcal{P})}$ | | | | | | | *89.7* | *88.7* | 92.5 | 89.6 | 92.6 | 92.0 | 92.4 | 89.8 |
| HM$_{(\mathcal{G}+\mathcal{C})}$ | | | | | | | | | *89.9* | *88.9* | 91.0 | 92.1 | 91.0 | 90.0 |
| HM$_{(\mathcal{P}+\mathcal{C})}$ | | | | | | | | | | | *89.6* | *88.5* | 90.9 | 92.0 |
| HM$_{(\mathcal{G}+\mathcal{P}+\mathcal{C})}$ | | | | | | | | | | | | | *90.5* | *89.7* |

Table 9: Upper bounds derived from combinations of two transliteration engines, where n-best=10

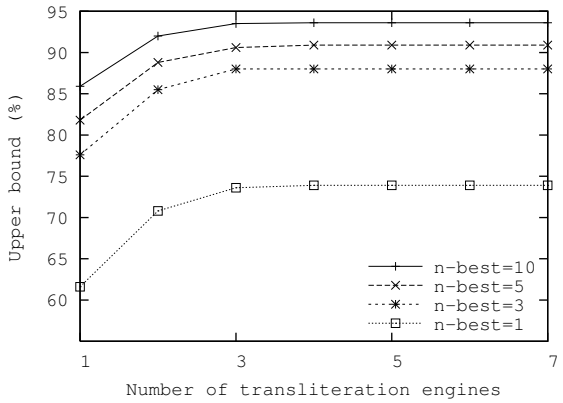


Figure 2: Effect of number of transliteration engines on upper bound in EJSet

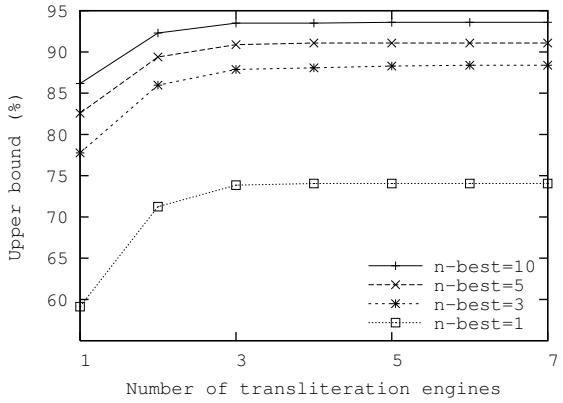


Figure 3: Effect of number of transliteration engines on upper bound in EKSet

thus have more chances of finding a correct transliteration hypothesis in the transliteration hypotheses they produced. Therefore, we could obtain better performance for both upper bound and our system with re-ranking if we used more transliteration engines. However, the performance converged at $|R| = 3$ regardless of $n$-best size. There has been a consensus that improvements expected from combinations of different systems are usually high when the systems are sufficiently different (Hoffmeister et al., 2007). For this reason, increasing $|R|$ by adding hybrid-model engines ($|R| \geq 4$) contributed little to increasing performance because the hybrid-model engines were based on three single-model engines; they thus often generated the same correct transliteration hypotheses as the single-model engines.

To investigate how different correct transliterations were produced by each engine, we compared the upper bound of $|R|$=1 and that of $|R|$=2 with every possible combination of two transliteration engines (see Table 9). Here, the upper bound of $|R|$=1 is represented as the combination between the same transliteration engine. According to our settings for $\alpha$, $\beta$, $\gamma$, $\delta_1$, $\delta_2$, and $\delta_3$, GM and HM or CM and HM produced similar correct transliterations — HM$_{(\mathcal{G}+\mathcal{P})}$'s and GM's in EJSet were similar to each other; while the other HMs' were similar to CM's in EJSet—and CM usually covered correct transliterations produced by PM. Therefore, the combination of HM$_{(\mathcal{G}+\mathcal{P})}$ and GM in EJSet showed the lowest improvement of upper bound among combinations of GM and the others; while the combination of GM and PM or GM and CM always showed significant improvement of the upper bound in both EJSet and EKSet.

## 5. Related Work

Researchers have developed transliteration systems based on different machine transliteration models — GM, PM, CM, and HM$_{(\mathcal{G}+\mathcal{P})}$ — over the last decade (Al-Onaizan and Knight, 2002; Bilac and Tanaka, 2004; Goto et al., 2003; Kang, 2001; Kang and Kim, 2000; Knight and Graehl, 1998; Oh and Choi, 2005). Even though individual transliteration engines have performed relatively well as can be seen from Table 4, multiple transliteration engines based on different transliteration models perform better than individual transliteration engines (Tables 4 and 5).

Al-Onaizan and Knight (2002), Qu et al. (2004), Zhang et al. (2005), and Oh et al. (2006b) used Web frequency acquired from either MKS, BKS, or BPS. However, as seen in Table 7, combinations of Web frequency acquired from the three Web search methods are better than using either MKS, BKS, or BPS alone in re-ranking hypotheses. Our system, using Web-frequency features produced by the three Web-search methods, performs better than the previous systems because MKS, BKS, and BPS enable Web frequency to be retrieved from different aspects.

Oh et al. (2006b) proposed a method of re-ranking transliterations, which combined two features simply. However, due to its simple framework, it had limitations in taking various features into account and combining them. Our SVM-based and MEM-based re-ranking methods make it

possible to effectively combine various features and thus improve transliteration (Table 5).

## 6. Conclusion

We described a machine transliteration system using multiple transliteration engines and hypothesis re-ranking. The use of multiple transliteration engines enables us to produce more transliteration hypotheses and to increase the chances of finding a correct transliteration hypothesis from these hypotheses. We used confidence-score, language-model, and Web-frequency features to train MEM-based and SVM-based re-ranking functions and used these to re-rank the hypotheses. Our experiments revealed that our re-ranking more effectively identified a correct transliteration hypothesis in transliteration hypotheses through using these various features than previous systems.

However, much work still remains to be done. We need to devise more effective features to re-rank hypotheses to enable our system to perform closer to upper bounds.

## 7. References


Y. Al-Onaizan and Kevin Knight. 2002. Translating named entities using monolingual and bilingual resources. In *Proc. of ACL '02*, pages 400–408.

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

Slaven Bilac and Hozumi Tanaka. 2004. Improving back-transliteration by combining information sources. In *Proc. of IJCNLP '04*, pages 542–547.

Michael Collins and Terry Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–70.

Hal Daume III and Daniel Marcu. 2004. NP bracketing by maximum entropy tagging and SVM reranking. In *Proc. of EMNLP*, pages 254–261.

J. Fiscus. 1997. A post-processing system to yield reduced word error rates: Recogniser Output Voting Error Reduction (ROVER). In *Proc. of IEEE Workshop on ASRU*, pages 347–352.

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

I. Goto, N. Kato, N. Uratani, and T. Ehara. 2003. Transliteration considering context information based on the maximum entropy method. In *Proc. of MT-Summit IX*, pages 125–132.

Bjorn Hoffmeister, Dustin Hillard, Stefan Hahn, Ralf Schulter, Mari Ostendorf, and Hermann Ney. 2007. Cross-site and intra-site asr system combination: Comparisons on lattice and 1-best methods. In *Procs. of ICASSP '07*.

Heng Ji, Cynthia Rudin, and Ralph Grishman. 2006. Re-ranking algorithms for name tagging. In *Proc. HLT/NAACL 06 Workshop on Computationally Hard Problems and Joint Inference in Speech and Language Processing*.

Thorsten Joachims. 2002. *Learning to Classify Text Using Support Vector Machines: Methods, Theory and Algorithms*. Kluwer Academic Publishers.

I. H. Kang and G. C. Kim. 2000. English-to-Korean transliteration using multiple unbounded overlapping phoneme chunks. In *Proc. of COLING '00*, pages 418–424.

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

Daisuke Kawahara and Sadao Kurohashi. 2006. Case frame compilation from the web using high-performance computing. In *Proc. of LREC '06*.

Kevin Knight and Jonathan Graehl. 1998. Machine transliteration. *Computational Linguistics*, 24(4):599 – 612.

Y. S. Nam. 1997. *Foreign dictionary*. Sung An Dang.

T. Nomoto. 2004. Multi-engine machine translation with voted language model. In *Proc. of ACL 2004*, pages 494–501.

Jong-Hoon Oh and Key-Sun Choi. 2002. An English-Korean transliteration model using pronunciation and contextual rules. In *Proc. of COLING2002*, pages 758–764.

Jong-Hoon Oh and Key-Sun Choi. 2005. An ensemble of grapheme and phoneme for machine transliteration. In *Proc. of IJCNLP '05*, pages 450–461.

Jong-Hoon Oh, Key-Sun Choi, and Hitoshi Isahara. 2006a. A comparison of different machine transliteration models. *Journal of Artificial Intelligence Research (JAIR)*, 27:119–151.

Jong-Hoon Oh, Key-Sun Choi, and Hitoshi Isahara. 2006b. Improving machine transliteration performance by using multiple transliteration models. In *Proc. of ICCPOL '06*, pages 85–96.

Yan Qu and Gregory Grefenstette. 2004. Finding ideographic representations of Japanese names written in Latin script via language identification and corpus validation. In *Proc. of ACL '04*, pages 183–190.

Richard Schwartz and Yen-Lu Chow. 1990. The N-best algorithm: An efficient and exact procedure for finding the N most likely sentence hypothesis. In *Procs. of ICASSP '90*, pages 81–84.

Libin Shen and Aravind K. Joshi. 2003. An SVM-based voting algorithm with application to parse reranking. In *Proc. of CoNLL '03*, pages 9–16.

Libin Shen, Anoop Sarkar, and Franz Josef Och. 2004. Discriminative reranking for machine translation. In *HLT-NAACL '04*, pages 177–184.

A. Stolcke. 2002. SRILM – an extensible language modeling toolkit. In *Proc. of Interspeech 2002*.

Ying Zhang, Fei Huang, and Stephan Vogel. 2005. Mining translations of OOV terms from the Web through cross-lingual query expansion. In *Proc. of SIGIR '05*, pages 669–670.