Latent Semantic Transliteration using Dirichlet Mixture

Masato Hagiwara **Satoshi Sekine**
Rakuten Institute of Technology, New York
215 Park Avenue South, New York, NY
{masato.hagiwara, satoshi.b.sekine}@mail.rakuten.com

Abstract

Transliteration has been usually recognized by spelling-based supervised models. However, a single model cannot deal with mixture of words with different origins, such as "get" in "piaget" and "target". Li et al. (2007) propose a class transliteration method, which explicitly models the source language origins and switches them to address this issue. In contrast to their model which requires an explicitly tagged training corpus with language origins, Hagiwara and Sekine (2011) have proposed the *latent class transliteration* model, which models language origins as latent classes and train the transliteration table via the EM algorithm. However, this model, which can be formulated as unigram mixture, is prone to overfitting since it is based on maximum likelihood estimation. We propose a novel *latent semantic transliteration* model based on Dirichlet mixture, where a Dirichlet mixture prior is introduced to mitigate the overfitting problem. We have shown that the proposed method considerably outperform the conventional transliteration models.

1 Introduction

Transliteration (e.g., バラクオバマ *baraku obama* "Barak Obama") is phonetic translation between languages with different writing systems, which is a major way of importing foreign words into different languages. Supervised, spelling-based grapheme-to-grapheme models such as (Brill and Moore, 2000; Li et al., 2004), which directly align characters in the training corpus without depending on phonetic information, and statistically computing their correspondence, have been a popular method to detect and/or generate transliterations, in contrast to phonetic-based methods such as (Knight and Jonathan, 1998). However, single, monolithic models fail to deal with sets of foreign words with multiple language origins mixed together. For example, the "get" part of "piaget / ピアジェ *piaje*" and "target / ターゲット *tāgetto*" differ in pronunciation and spelling correspondence depending on their source languages, which are French and English in this case.

To address this issue, Li et al. (2007) have proposed *class transliteration model*, which explicitly models and classifies classes of languages (such as Chinese Hanzi, Japanese Katakana, and so on) and genders, and switches corresponding transliteration models based on the input. This model requires training sets of transliterated word pairs tagged with language origin, which is difficult to obtain. Hagiwara and Sekine proposed the *latent class transliteration (LCT)* model (Hagiwara and Sekine, 2011), which models source language origins as directly unobservable latent classes and applies appropriate transliteration models to given transliteration pairs. The model parameters are learned from corpora without language origins in an unsupervised manner. This enables us to correctly assign latent classes for English and French to "piaget / ピアジェ*piaje*" and "target / ターゲッ

Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, pages 30–37,
Jeju, Republic of Korea, 8-14 July 2012.

⊦ *tāgetto*" and to identify their transliteration correspondence correctly. However, this model is based on maximum likelihood estimation on multinomials and thus sensitive to noise in the training data such as transliteration pairs with irregular pronunciation, and tends to overfit the data.

Considering the atomic re-writing unit (transliteration unit, or TU, e.g., "get / ゲッ ⊦ *getto*") as a word, and a transliteration pair as a document consisting of a word sequence, class-based transliteration can be modeled by the perfect analogy to document topic models proposed in tha past. In fact, the LCT model, where the transliteration probability is defined by a mixture of multinomials, can be regarded as a variant of a topic model, namely Unigram Mixuture (UM) (Nigam et al., 2000). There has been an extension of unigram mixture proposed (Sjölander et al., 1996; Yamamoto and Sadamitsu, 2005) which introduces a Dirichlet mixture distribution as a prior and alleviates the overfitting problem. We can expect to improve the transliteration accuracy by formulating the transliteration problem using a similar framework to these topic models.

In this paper, we formalize class-based transliteration based on language origins in the framework of topic models. We then propose the *latent semantic transliteration* model based on Dirichlet mixture (DM-LST). We show through experiments that it can significantly improve the transliteration performance by alleviating the overfitting issue.

Note that we tackle the task of *transliteration generation* in this paper, in contrast to *transliteration recognition*. A transliteration generation task is, given an input word $\boldsymbol{s}$ (such as "piaget"), the system is asked to generate from scratch the most probable transliterated word $\boldsymbol{t}$ (e.g., "ピアジェ *piaje*"). The transliteration recognition task, on the other hand, is to induce the most probable transliteration $\boldsymbol{t}^* \in T$ such that $\boldsymbol{t}^* = \arg\max_{\boldsymbol{t} \in T} P(\langle \boldsymbol{s}, \boldsymbol{t} \rangle)$ given the input word $\boldsymbol{s}$ and a pool of transliteration candidates T. We call $P(\langle \boldsymbol{s}, \boldsymbol{t} \rangle)$ *transliteration model* in this paper.

This model can be regarded as the hybrid of an unsupervised alignment technique for transliteration and class-based transliteration. Related researches for the former include (Ahmad and Kondrak, 2005), who estimate character-based error probabilities from query logs via the EM algorithm. For the latter, Llitjos and Black (2001) showed that source language origins may improve the pronunciation of proper nouns in text-to-speech systems.

The structure of this paper is as follows: we introduce the alpha-beta model(Brill and Moore, 2000) in Section 2, which is the most basic spelling-based transliteration model on which other models are based. In the following Section 3, we introduce and relate the joint source channel (JSC) model (Li et al., 2004) to the alpha-beta model. We describe the LCT model as an extension to the JSC model in Section 4. In Section 5, we propose the DM-LST model, and show the experimental results on transliteration generation in Section 6.

2 Alpha-Beta Model

In this section, we describe the alpha-beta model, which is one of the simplest spelling-based transliteration models. Though simple, the model has been shown to achieve better performance in tasks such as spelling correction (Brill and Moore, 2000), transliteration (Brill et al., 2001), and query alteration (Hagiwara and Suzuki, 2009).

The method directly models spelling-based re-writing probabilities of transliteration pairs. It is an extension to the normal edit distance, where the cost of operations (substitution, insertion, and deletion) is fixed to 1, and assigns a probability to a string edit operation of the form $s_i \rightarrow t_i$ (s_i and t_i are any substrings of length 0 to w). We call the unit operation of string re-writing $u_i = \langle s_i, t_i \rangle$ as transliteration unit (TU) as in (Li et al., 2004). The total transliteration probability of re-writing a word $\boldsymbol{s}$ to $\boldsymbol{t}$ is given by

$$P_{AB}(\langle \boldsymbol{s}, \boldsymbol{t} \rangle) = \max_{u_1 \ldots u_f} \prod_{i=1}^{f} P(u_i), \qquad (1)$$

where f is the number of TUs and $u_1 ... u_f$ is any sequence of TUs (e.g., "pi ／ピ゜ a ／ア get ／

ジェ") created by splitting up the input/output transliteration pair $\langle \boldsymbol{s}, \boldsymbol{t} \rangle$. The above equation can be interpreted as a problem of finding a TU sequence $u_1 ... u_f$ which maximizes the probability defined by the product of individual probabilities of independent TUs. After taking the logarithm of the both sides, and regarding $-\log P(u_i)$ as the cost of string substitution $s_i \rightarrow t_i$, the problem is equivalent to minimizing the sum of re-writing costs, and therefore can be efficiently solved by dynamic programming as done in the normal edit distance.

TU probabilities $P(u_i)$ are calculated from a training set of transliteration pairs. However, training sets usually lack alignment information specifying which characters in $\boldsymbol{s}$ corresponding which characters in $\boldsymbol{t}$. Brill and Moore (2000) resorted to heuristics to align same characters and to induce the alignment of string chunks. Hagiwara and Sekine (2011) converted Japanese Katakana sequences into Roman alphabets because their model also assumed that the strings s_i and t_i are expressed in the same alphabet system. Our method on the contrary, does not pose such assumption so that strings in different writing systems (such as Japanese Katakana and English alphabets, and Chinese characters and English alphabets, etc.) can be aligned without being converted to phonetic representation. For this reason, we cannot adopt algorithms (such as the one described in (Brill and Moore, 2000)) which heuristically infer alignment based on the correspondence of the same characters.

When applying this alpha-beta model, we computed TU probabilities by counting relative frequencies of all the alignment possibilities for a transliteration pair. For example, all the alignment possibilities for a pair of strings "abc" and "xy" are (a-x b-y c-ε), (a-x b-ε c-y), and (a-ε b-x c-y). By considering merging up to two adjacent aligned characters in the first alignment, one obtains the following five aligned string pairs: a-x, b-y, c-ε, ab-xy bc-y. Note that all the transliteration models described in this paper implicitly depend on the parameter w indicating the maximum length of character n-grams. We fixed $w = 3$ throughout this paper.

3 Joint Source Channel Model

The alpha-beta model described above has shortcomings that the character alignment is fixed based on heuristics, and it cannot capture the dependencies between TUs. One example of such dependencies is the phenomenon that the suffix "-ed" in English verbs following a voiced consonant is pronounced /d/, whereas the one followed by an unvoiced consonant is /t/. This section describes the JSC model(Li et al., 2004), which was independently proposed from the alpha-beta model. The JSC model is essentially equivalent to the alpha-beta model except: 1) it can also incorporate higher order of n-grams of TUs and 2) the TU statistics is taken not by fixing the heuristic initial alignment but by iteratively updating via an EM-like algorithm.

In the JSC model, the transliteration probability is defined by the n-gram probabilities of TUs $u_i = \langle s_i, t_i \rangle$ as follows:

$$P_{JSC}(\langle \boldsymbol{s}, \boldsymbol{t} \rangle) = \prod_{i=1}^{f} P(u_i | u_{i-n+1}, ..., u_{i-1}). \quad (2)$$

Again, f is the number of TUs. The TU n-gram probabilities $P(u_i | u_{i-n+1}, ..., u_{i-1})$ can be calculated by the following iterative updates similar to the EM algorithm:

1. Set the initial alignment randomly.
2. E-step: Take the TU n-gram statistics fixing the current alignment, and update the transliteration model.
3. M-step: Compute the alignment based on the current transliteration model. The alignment is inferred by dynamic programming similar to the alpha-beta model.
4. Iterate the E- and M- step until convergence.

Notice the alpha-beta model and the JSC model are both transliteration recognition models. In order to output a transliterated word $\boldsymbol{t}$ for a given input $\boldsymbol{s}$, we generated transliteration candidates with high probability using a stack

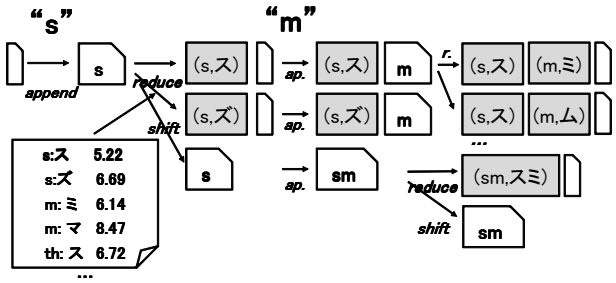

Figure 1: Overview of the stack decoder (generation of "スミス *sumisu*" from the input "smith")

decoder, whose overview is shown in Figure 1. One character in the input string $\boldsymbol{s}$ (which is "smith" in the figure) is given at a time, which is appended at the end of the last TUs for each candidate. (the *append* operation in the figure). Next, the last TU of each candidate is either *reduced* or *shifted*. When it is *reduced*, top R TUs with highest probabilities are generated and fixed referring to the TU table (shown in the bottom-left of the figure). In Figure 1, two candidates, namely ("s", "ス *su*") and ("s", "ズ *zu*") are generated after the character "s" is given. When the last TU is *shifted*, it remains unchanged and unfixed for further updates. Every time a single character is given, the transliteration probability is computed using Eq. 2 for each candidate, and all but the top-B candidates with highest probabilities are discarded. The reduce width R and the beam width B were determined using the determined using development sets, as mentioned in Section 6.

4 Latent Class Transliteration Model

As mentioned in Section 1, the alpha-beta model and the JSC model build a single transliteration model which is simply the monolithic average of training set statistics, failing to capture the difference in the source language origins. Li et al. (2004) address this issue by defining classes c, i.e., the factors such as source language origins, gender, and first/last names, etc. which affect the transliteration probability. The authors then propose the class transliteration model which gives the probability of $\boldsymbol{s} \rightarrow \boldsymbol{t}$ as follows:

$$P_{LI}(\boldsymbol{t}|\boldsymbol{s}) = \sum_c P(\boldsymbol{t}, c|\boldsymbol{s}) = \sum_c P(c|\boldsymbol{s}) P(\boldsymbol{t}|c, \boldsymbol{s}) \quad (3)$$

However, this model requires a training set explicitly tagged with the classes. Instead of assigning an explicit class c to each transliterated pair, Hagiwara and Sekine (2011) introduce a random variable z which indicates implicit classes and conditional TU probability $P(u_i|z)$. The latent class transliteration (LCT) model is then defined as[1]:

$$P_{\mathrm{LCT}}(\langle \boldsymbol{s}, \boldsymbol{t} \rangle) = \sum_{z=1}^{K} P(z) \prod_{i=1}^{f} P(u_i|z) \quad (4)$$

where K is the number of the latent classes. The latent classes z correspond to classes such as the language origins and genders mentioned above, shared by sets of transliterated pairs with similar re-writing characteristics. The classes z are not directly observable from the training set, but can be induced by maximizing the training set likelihood via the EM algorithm as follows.

$$\textbf{Parameters:} \quad P(z=k) = \pi_k, \ P(u_i|z) \quad (5)$$

$$\textbf{E-Step:} \ \gamma_{nk} = \frac{\pi_k P(\langle \boldsymbol{s}_n, \boldsymbol{t}_n \rangle | z=k)}{\sum_{k'=1}^{K} \pi_{k'} P(\langle \boldsymbol{s}_n, \boldsymbol{t}_n \rangle | z=k')}, \quad (6)$$

$$P(\langle \boldsymbol{s}_n, \boldsymbol{t}_n \rangle | z) = \max_{u_1 .. u_f} \prod_{i=1}^{f_n} P(u_i|z) \quad (7)$$

$$\textbf{M-Step:} \quad \pi_k^{new} \propto \sum_{n=1}^{N} \gamma_{nk}, \quad (8)$$

$$P(u_i|z=k)^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{nk} \frac{f_n(u_i)}{f_n} \quad (9)$$

where $N_k = \sum_n \gamma_{nk}$. Here, $\langle \boldsymbol{s}_n, \boldsymbol{t}_n \rangle$ is the n-th transliterated pair in the training set, and f_n and $f_n(u_i)$ indicate how many TUs there are in total in the n-th transliterated pair, and how many times the TU u_i appeared in it, respectively. As done in the JSC model, we update the alignment in the training set before the E-Step for each iteration. Thus f_n takes different values

[1]Note that this LCT model is formalized by introducing a latent variable to the transliteration generative probability $P(\langle \boldsymbol{s}, \boldsymbol{t} \rangle)$ as in the JSC model, not to $P(\boldsymbol{t}|\boldsymbol{s})$.

from iteration to iteration in general. Furthermore, since the alignment is updated based on $P(u_i|z)$ for each $z = k$, M different alignment candidates are retained for each transliterated pairs, which makes the value of f_n dependent on k, i.e., f_n^k. We initialize $P(z = k) = 1/M$ to and $P(u_i|z) = P_{AB}(u) + \varepsilon$, that is, the TU probability induced by the alpha-beta algorithm plus some random noise ε.

Considering a TU as a word, and a transliteration pair as a document consisting of a word sequence, this LCT model defines the transliteration probability as the mixture of multinomials defined over TUs. This can be formulated by unigram mixture (Nigam et al., 2000), which is a topic model over documents. This follows a generation story where documents (i.e., transliterated pairs) are generated firstly by choosing a class z by $P(z)$ and then by generating a word (i.e., TU) by $P(u_i|z)$. Nevertheless, as mentioned in Section 1, since this model trains the parameters based on the maximum likelihood estimation over multinomials, it is vulnerable to noise in the training set, thus prone to overfit the data.

5 Latent Semantic Transliteration Model based on Dirichlet Mixture

We propose the latent semantic transliteration model based on Dirichlet mixture (DM-LST), which is an extension to the LCT model based on unigram mixture. This model enables to prevent multinomials from being exceedingly biased towards the given data, still being able to model the transliteration generation by a mixture of multiple latent classes, by introducing Dirichlet mixture as a prior to TU multinomials. The compound distribution of multinomials when their parameters are given by Dirichlet mixtures is given by the Polya mixture distribution(Yamamoto and Sadamitsu, 2005):

$$P_{DM}(\langle \boldsymbol{s}, \boldsymbol{t} \rangle) \tag{10}$$

$$= \int P_{Mul}(\langle \boldsymbol{s}, \boldsymbol{t} \rangle; \boldsymbol{p}) P_{DM}(\boldsymbol{p}; \boldsymbol{\lambda}, \boldsymbol{\alpha}_1^K) d\boldsymbol{p}$$

$$\propto \sum_{k=1}^{K} \lambda_k P_{Polya}(\langle \boldsymbol{s}, \boldsymbol{t} \rangle; \boldsymbol{\alpha}_1^K) \tag{11}$$

$$= \sum_{k=1}^{K} \lambda_k \frac{\Gamma(\alpha_k)}{\Gamma(\alpha_k + f)} \prod_{i=1}^{f} \frac{\Gamma(f(u_i) + \alpha_{ku_i})}{\Gamma(\alpha_{ku_i})}$$

where $P_{Mul}(*; \boldsymbol{p})$ is multinomial with the parameter $\boldsymbol{p}$. P_{DM} is Dirichlet mixture, which is a mixture (with co-efficients $\lambda_1, ..., \lambda_K$) of K Dirichlet distributions with parameters $\alpha_1^K = (\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, ..., \boldsymbol{\alpha}_K)$.

The model parameters can be induced by the following EM algorithm. Notice that we adopted a fast induction algorithm which extends an induction method using leaving-one-out to mixture distributions(Yamamoto et al., 2003).

$$\textbf{Parameters:} \quad \boldsymbol{\lambda} = (\lambda_1, ..., \lambda_K), \tag{12}$$

$$\boldsymbol{\alpha}_1^K = (\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, ..., \boldsymbol{\alpha}_K) \tag{13}$$

$$\textbf{E-Step:} \quad \eta_{nk} = \frac{\lambda_k P_{Polya}(\langle \boldsymbol{s}_n, \boldsymbol{t}_n \rangle; \boldsymbol{\alpha}_k)}{\sum_{k'} \lambda_{k'} P_{Polya}(\langle \boldsymbol{s}_n, \boldsymbol{t}_n \rangle; \boldsymbol{\alpha}_{k'})} \tag{14}$$

$$\textbf{M-Step:} \quad \lambda_k^{new} \propto \sum_{n=1}^{N} \eta_{nk} \tag{15}$$

$$\alpha_{ku}^{new} = \alpha_{ku} \frac{\sum_n \eta_{nk} \{f_n(u)/(f_n(u) - 1 + \alpha_{ku})\}}{\sum_n \eta_{nk} \{f_n/(f_n - 1 + \alpha_k)\}} \tag{16}$$

The prediction distribution when a single TU u is the input is given $P_{DM}(u) = \sum_{k=1}^{K} \lambda_k \alpha_{ku}/\alpha_k$. We therefore updated the alignment in the training corpus, as done in the JSC model updates, based on the probability proportional to α_{ku}/α_k for each k before every M-Step. The parameters are initially set to $\lambda_k = 1/K$, $\alpha_{ku} = P_{AB}(u) + \varepsilon$, as explained in the previous section.

Since neither LCT nor DM-LST is a transliteration generation model, we firstly generated transliteration candidates T by using the JSC model and the stack decoder (Section 3) as a

baseline, then re-ranked the candidates using the probabilities given by LCT (Eq. 4 or DM-LST (Eq. 11), generating the re-ranked list of transliterated outputs. Because the parameters trained by the EM algorithm differ depending on the initial values, we trained 10 models $P^1_{DM}, ..., P^{10}_{DM}$ using the same training data and random initial values and computed the average $\frac{1}{10}\sum_{j=1}^{10} P^j_{DM}(\langle \boldsymbol{s}, \boldsymbol{t} \rangle)$ to be used as the final transliteration model.

It is worth mentioning that another topic model, namely latent Dirichlet allocation (LDA) (Blei et al., 2003), assumes that words in a document can be generated from different topics from each other. This assumption corresponds to the notion that TUs in a single transliterated pairs can be generated from different source languages, which is presumably a wrong assumption for transliteration tasks, probably except for compound-like words with mixed origins such as "naïveness". In fact, we confirmed through a preliminary experiment that LDA does not improve the transliteration performance over the baseline.

6 Experiments

6.1 Evaluation

In this section, we compare the following models: alpha-beta (AB), joint source channel (JSC), latent class transliteration (LCT), and latent semantic transliteration based on Dirichlet mixture (DM-LST).

For the performance evaluation, we used three language pairs, namely, English-Japanese (En-Ja), English-Chinese (En-Ch), and English-Korean (En-Ko), from the transliteration shared task at NEWS 2009 (Li et al., 2009a; Li et al., 2009b). The size of each training/test set is shown in the first column of Table 1. In general, $\boldsymbol{r}_n$, a set of one or more reference transliterated words, is associated with the n-th input $\boldsymbol{s}_n$ in the training/test corpus. Let $c_{n,i}, c_{n,2}, ...$ be the output of the transliteration system, i.e., the candidates with highest probabilities assigned by the transliteration model being evaluated. We used the following three performance measures:

· **ACC** (averaged Top-1 accuracy): For every $\langle \boldsymbol{s}_n, \boldsymbol{r}_n \rangle$, let a_n be $a_n = 1$ if the candidate with the highest probability $c_{n,1}$ is contained in the reference set $\boldsymbol{r}_n$ and $a_n = 0$ otherwise. ACC is then calculated as $ACC \frac{1}{N}\sum_{i=1}^{N} s_n$.

· **MFS** (mean F score): Let the reference transliterated word closest to the top-1 candidate $c_n, 1$ be $r^*_n = \arg\min_{r_{n,j} \in \boldsymbol{r}_n} ED(c_{n,1}, r_{n,j})$, where ED is the edit distance. The F-score of the top candidate $c_{n,1}$ for the n-th input $\boldsymbol{s}_n$ is then given by:

$$P_n = LSC(c_{n,1}, r^*_n)/|c_{n,1}| \quad (17)$$
$$R_n = LCS(c_{n,1}, r^*_n)/|r^*_n| \quad (18)$$
$$F_n = 2R_iP_i/(R_i + P_i), \quad (19)$$

where $|x|$ is the length of string x, and $LCS(x, y)$ is the length of the longest common subsequence of x and y. Edit distance, lengths of strings, and LCS are measured in Unicode characters. Finally, MFS is defined as $MFS = \frac{1}{N}\sum_{i=1}^{N} F_n$.

· **MRR** (mean reciprocal rank): Of the ranked candidates $c_{n,1}, c_{n,2}, ...$, let the highest ranked one which is also included in the reference set $\boldsymbol{r}_n$ be $c_{n,j}$. We then define reciprocal rank $RR_n = 1/j$. If none of the candidates are in the reference, $RR_n = 0$. MRR is then defined by $MRR = \frac{1}{N}\sum_{n=1}^{N} RR_n$.

We used Kneser-Nay smoothing to smooth the TU probabilities for LCT. The number of EM iterations is fixed to 15 for all the models, based on the result of preliminary experiments.

The reduce width R and the beam width B for the stack decoder are fixed to $R = 8$ and $B = 32$, because the transliteration generation performance increased very little beyond these widths based on the experiment using the development set. We also optimized M, i.e., the number of latent classes for LCT and DM-LST, for each language pair and model in the same way based on the development set.

Table 1: Performance comparison of transliteration models

Language pair	Model	ACC	MFS	MRR
En-Ja	AB	0.293	0.755	0.378
Train: 23,225	JSC	0.326	0.770	0.428
Test: 1,489	LCT	0.345	0.768	0.437
	DM-LST	**0.349**	**0.776**	**0.444**
En-Ch	AB	0.358	0.741	0.471
Train: 31,961	JSC	0.417	0.761	0.527
Test: 2,896	LCT	0.430	0.764	0.532
	DM-LST	**0.445**	**0.770**	**0.546**
En-Ko	AB	0.145	0.537	0.211
Train: 4,785	JSC	0.151	0.543	0.221
Test: 989	LCT	0.079	0.483	0.167
	DM-LST	**0.174**	**0.556**	**0.237**

6.2 Results

We compared the performance of each transliteration model in Table 1. For the language pairs En-Ja and En-Ch, all the performance increase in the order of AB < JSC < LCT < DM-LST, showing the superiority our proposed method. For the language pair En-Ko, the performance for LCT re-ranking considerably decreases compared to JSC. We suspect this is due to the relatively small number of training set, which caused the excessive fitting to the data. We also found out that the optimal value of M which maximizes the performance of DM-LST is equal to or smaller than that of LCT. This goes along with the findings (Yamamoto and Sadamitsu, 2005) that Dirichlet mixture often achieves better language model perplexity with smaller dimensionality compared to other models.

Specific examples in the En-Ja test set whose transliteration is improved by the proposed methods include "dijon ／ディジョン *dijon*" and "goldenberg ／ゴールデンバーグ *gōrudenbāgu*". Conventional methods, including LCT, suggested "ディヨン *diyon*" and "ゴールデンベルグ *gōrudenberugu*", meaning that the transliteration model is affected and biased towards non-English pronunciation. The proposed method can retain the major class of transliteration characteristics (which is English in this case) and can deal with multiple language origins depending on transliteration pairs at the same time.

This trend can be also confirmed in other language pairs, En-Ch and En-Ko. In En-Ch, the transliterated words of "covell" and "netherwood" are improved " "科夫尔 *kefuer* →科维尔 *keweier*" and "内特赫伍德 *neitehewude* →内瑟伍德 *neisewude*", respectively. in En-Ko, the transliterated word of "darling" is improved "다르링 *dareuling*" → "달링 *dalling*".

We also observed that "gutheim / 古特海姆 *gutehaimu* in En-Ch and martina / 마르티나 *mareutina* in En-Ko are correctly translated by the proposed method, even though they do not have the English origin. Generally speaking, however, how these non-English words are pronounced depend on the context, as "charles" has different pronunciation in English and in French, with the soft "sh" sound at the beginning. We need external clues to disambiguate such transliteration, such as context information and/or Web statistics.

7 Conclusion

In this paper, we proposed the latent semantic transliteration model based on Dirichlet mixture (DM-LST) as the extension to the latent class transliteration model. The experimental results showed the superior transliteration performance over the conventional methods, since DM-LST can alleviate the overfitting problem and can capture multiple language origins. One drawback is that it cannot deal with dependencies of higher order of TU n-grams than bigrams. How to incorporate these dependencies into the latent transliteration models is the future work.

References

Farooq Ahmad and Grzegorz Kondrak. 2005. Learning a spelling error model from search query logs. In *Proc. of EMNLP-2005*, pages 955–962.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Eric Brill and Robert C. Moore. 2000. An improved error model for noisy channel spelling. In *Proc. ACL-2000*, pages 286–293.

Eric Brill, Gary Kacmarcik, and Chris Brockett. 2001. Automatically harvesting katakana-english term pairs from search engine query logs. In *Proc. NLPRS-2001*, pages 393–399.

Masato Hagiwara and Satoshi Sekine. 2011. Latent class transliteration based on source language origin. In *Proc. of ACL-HLT 2011*, pages 53–57.

Masato Hagiwara and Hisami Suzuki. 2009. Japanese query alteration based on semantic similarity. In *Proc. of NAACL-2009*, page 191.

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

Haizhou Li, Zhang Min, and Su Jian. 2004. A joint source-channel model for machine transliteration. In *Proc. of ACL 2004*, pages 159–166.

Haizhou Li, Khe Chai Sum, Jin-Shea Kuo, and Minghui Dong. 2007. Semantic transliteration of personal names. In *Proc. of ACL 2007*, pages 120–127.

Haizhou Li, A Kumaran, Vladimir Pervouchine, and Min Zhang. 2009a. Report of news 2009 machine transliteration shared task. In *Proc. of the 2009 Named Entities Workshop*, pages 1–18.

Haizhou Li, A Kumaran, Min Zhang, and Vladimir Pervouchine. 2009b. Whitepaper of news 2009 machine transliteration shared task. In *Proc. of the 2009 Named Entities Workshop*, pages 19–26.

Ariadna Font Llitjos and Alan W. Black. 2001. Knowledge of language origin improves pronunciation accuracy. In *Proc. of Eurospeech*, pages 1919–1922.

Kamal Nigam, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell. 2000. Text classification from labeled and unlabeled documents using em. *Machine Learning*, 39(2):103–134.

K. Sjölander, K. Karplus, M. Brown, R. Hunghey, A. Krogh, I.S. Mian, and D. Haussler. 1996. Dirichlet mixtures:a method for improved detection of weak but significant protein sequence homology. *Computer Applications in the Biosciences*, 12(4):327–345.

Mikio Yamamoto and Kugatsu Sadamitsu. 2005. Dirichlet mixtures in text modeling. *CS Technical Report*, CS-TR-05-1.

Mikio Yamamoto, Kugatsu Sadamitsu, and Takuya Mishina. 2003. Context modeling using dirichlet mixtures and its applications to language models (in japnaese). *IPSJ*, 2003-SLP-48:29–34.