10-2004

Improving transliteration with precise alignment of phoneme chunks and using contextual features

Wei GAO
Singapore Management University, weigao@smu.edu.sg

Kam-Fai WONG

Wai LAM

Improving Transliteration with Precise Alignment of Phoneme Chunks and Using Contextual Features

Wei Gao, Kam-Fai Wong, and Wai Lam

Department of Systems Engineering and Engineering Management,
The Chinese University of Hong Kong,
Shatin, N.T., Hong Kong, China
{wgao, kfwong, wlam}@se.cuhk.edu.hk

Abstract. Automatic transliteration of foreign names is basically regarded as a diminutive clone of the machine translation (MT) problem. It thus follows IBM's conventional MT models under the source-channel framework. Nonetheless, some parameters of this model dealing with *zero-fertility* words in the target sequences, can negatively impact transliteration effectiveness because of the inevitable *inverted* conditional probability estimation. Instead of source-channel, this paper presents a *direct* probabilistic transliteration model using contextual features of phonemes with a tailored alignment scheme for phoneme chunks. Experiments demonstrate superior performance over the source-channel for the task of English-Chinese transliteration.

1 Introduction

Automatic transliteration of foreign names, e.g. names of people, places, organizations, etc., is recognized as an important issue in many cross language applications. Cross-lingual information retrieval involves query keyword translation from the source to target language and document translation in the opposite direction. For similar reasons, machine translation and spoken language processing, such as cross-lingual spoken document retrieval and spoken language translation, also encounters the same problem of translating proper names. Contemporary lexicon-based translation is ineffective as proper name dictionaries can never be comprehensive. New names appear almost daily and become unregistered vocabulary in the lexicon. This is known as the *Out-Of-Vocabulary* (OOV) problem. The lack of translation for OOV names can impact the performance of applications adversely and sometimes seriously.

Based on pronunciations, foreign names can usually be translated, or more appropriately transliterated, into target languages, which were originally hand-coded with rules of thumb by human translators. De facto standard has been established, but is often inconsistently used. The rules are subjected to the interpretation of individual producers. In Mandarin Chinese, for instance, the name

S. H. Myaeng et al. (Eds.): AIRS 2004, LNCS 3411, pp. 106–117, 2005.

of "Bin Laden" can be translated as /ben la deng/[1], /bin la deng/, /ben la dan/ and /bin la dan/. Sometimes dialectical features can further ambiguate the standard. For these reasons, rule-based transliteration approach has been undermined in English-to-Chinese machine translation applications. Thus, an effective data-driven transliteration model is required.

In this paper, we present a statistical phoneme-based method for forward transliteration of foreign names from English to Chinese. Grapheme-to-phoneme transformation and Pinyin-to-Hanzi conversion applied in the phoneme-based methods are extensively studied areas. We focus on the intermediate tasks for transliterating phoneme pairs. The rest of the paper is organized as follows: Section 2 summarizes related work; Section 3 explains the drawbacks of source-channel model in our task; Section 4 illustrates our methods in detail; Section 5 presents and analyses experimental results; Section 6 concludes this paper.

2 Related Work

Several approaches have been proposed for automatic name transliteration between various language pairs. Based on the source-channel framework, [7] described a generative model, in which they adopted finite state transducers and a channel decoder to transform transliterated names in Japanese back to their origins in English. The source-channel model was later on applied or extended by a number of other tasks: backward transliteration from Arabic to English in [12], forward transliteration from English to Korean in [8], and forward transliteration from English to Chinese in [13].

The aim of work by [13] is closest to ours. Their fundamental equation derived from the IBM statistical machine translation (SMT) model proposed by [2]:

$$\hat{C} = \underset{C}{\operatorname{argmax}}\, p(C|E) = \underset{C}{\operatorname{argmax}}\, \{p(E|C) \times p(C)\} \tag{1}$$

where $E = e_1^{|E|}$ denotes a $|E|$-phoneme English word as the observation on channel output, and $C = c_1^{|C|}$ represents E's $|C|$-phoneme Chinese translation by pinyin as the source of channel input. As shown in Fig. 1, the channel decoder reverses the direction to find the most probable input pinyin sequence $\hat{C}$ given an observation E. The posterior probability $p(C|E)$ is indirectly maximized by optimizing the combination of the transliteration model $p(E|C)$ and the language model $p(C)$. $p(E|C)$ was trained from name pairs represented by International Phonetic Alphabet (IPA) symbols for English names (obtained from a speech synthesis engine) and pinyin notations for their Chinese counterparts (obtained from a Hanzi-Pinyin dictionary). It proceeded by Expectation-Maximization (EM) iterations of standard IBM SMT model training method by using GIZA++ toolkit, bootstrapping from Model-1 through Model-4 [13]. Language model $p(C)$ was trained by using pinyin symbol trigrams and applying

[1] Mandarin pinyin is used as phonetic representation of Chinese characters throughout this paper. For simplicity, we ignore the four tones in the pinyin system.

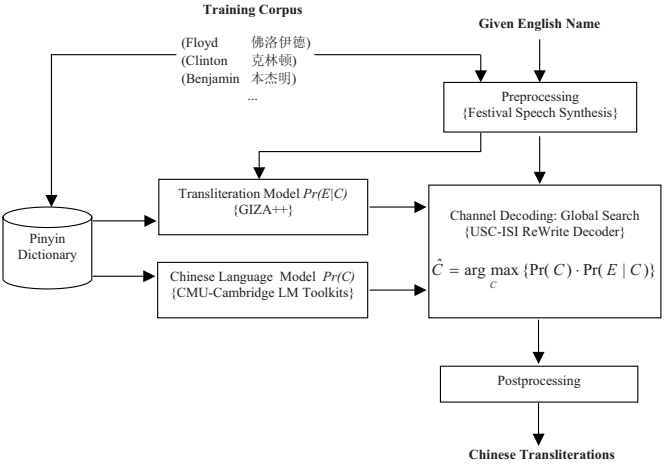

Fig. 1. English-to-Chinese transliteration system [13] based on IBM SMT model

Good-Turing smoothing with *Katz back-off* in CMU-Cambridge language modeling toolkits [3]. Search was done by use of USC-ISI ReWrite Decoder [5]. The method demonstrates pinyin error rates in edit distance of around 50% [13].

3 Drawbacks of Source-Channel

When the IBM SMT model is applied in our task, i.e. English-to-Chinese transliteration, it has several limitations:

1. $p(E|C)$ is approximated by the Markov chains [11] under Markov assumption (zero order or first-order) on state transition as well as conditional independence assumption on observation. Markov assumption hypothesizes that the transition probability to a state, i.e. phoneme of Chinese pinyin, depends only on its previous one state at most. Longer history may suffer from data sparseness and renders the model computationally expensive with the increase of state space; conditional independence assumption assumes that an observation unit, i.e. English phoneme, depends only on the state that generates it, not on its neighboring observation units. With these hypotheses, it is hard to extend the model by using additional dependencies, such as flexible features of neighboring phonemes. Albeit the trigram language model $p(C)$ is combined, (1) cannot be be optimal unless both $p(E|C)$ and $p(C)$ use the true probability distributions. Yet, the used models and training methods in machine translation empirically testified that they only provided poor approximations of the true distributions [9, 13].
2. Because of the *inverted* conditional probability $p(E|C)$, only a target language phoneme can be associated with a contiguous group of source language phonemes, but not vice visa, i.e. never could one English phoneme be con-

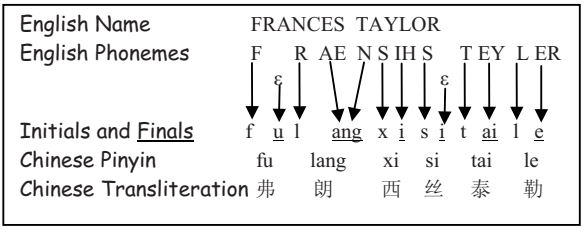

Fig. 2. English-to-Chinese transliteration example in [13], considering unaligned symbols as zero-fertility

verted to a group of pinyin symbols. The example in [13] exposes this obvious limitation (see Fig. 2[2]): /u/ and the second /i/ in the third line have to be considered as spuriously produced from nothing or from a mute ε. Under the IBM models, such inserted symbols are known as *zero-fertility* "words". They are deleted by source-channel during training and reproduced by decoder by considering inserting one of them before each target symbol of each remaining unaligned source phoneme in terms of the number of possible zero-fertility symbols [5]. Although adding zero-fertility symbols may increase the probability of hypothesis, incorrect transliterations are still abundant as such insertions are frequent.

3. Due to smoothing, the language model may not assign zero probability to an illegal pinyin sequence, e.g. one containing two consecutive initials [13]. Such sequences need to be manually corrected by inserting certain finals between them until a legitimate pinyin sequence is obtained. Moreover, the training of language model is independent of transliteration model and their combination sometimes can yield unpredictable results.

4 Direct Transliteration Model

Instead of source-channel, we aim to estimate the posterior probability directly. We rectify the angle of observation to avoid the use of the reversed conditional probability. Figure 3 shows the application of the alignment scheme of our approach to the previous example in Fig. 2, where we look possible combinations of pinyin symbols as initial-final clusters converted from single English phonemes. The condition of the probability to be estimated thus turns out to be E instead of C. The distribution can be approximated directly by Maximum Entropy (*MaxEnt*) approach [1]. Under *MaxEnt* model, the language model can be considered as an optional feature [9]. Its absence could be compensated if other cutting edge features could be chosen.

[2] Lowercase letters denote pinyin symbols. Capital letters are English phonemes represented by computer-readable IPA notations — ARPABET.

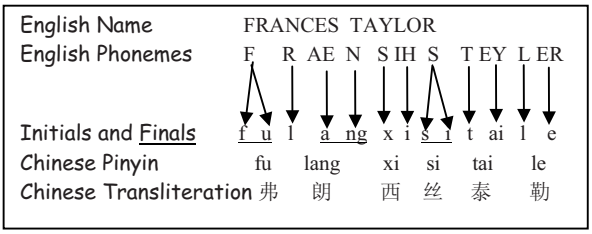

Fig. 3. Our phoneme alignment scheme in direct transliteration modeling

4.1 Baseline of Direct Model

We introduce the pinyin mapping units of each e_i denoted by cmu_i, which can be individual pinyin symbols or clusters of initial and final. In an alignment, each English phoneme aligns to only one *cmu*. Thus, the transliteration model $p(C|E)$ can be approximated by:

$$p(C|E) \approx \prod_{i=1}^{|E|} p(cmu_i|e_i) \quad . \tag{2}$$

The unknown *cmu*s (clusters) can be discovered on the fly during EM training for computing the *Viterbi* alignments and symbol-mapping probabilities by using GIZA++, where we can make the source and target consistent with the actual transliteration direction, i.e. from English to Chinese.

Equation (2) gives poor approximation as no contextual feature is considered. From our perspective, the transliteration is to classify each phoneme of a given English name into its most probable *cmu* according to the frames of various features. We then yield a better approximation:

$$p(C|E) \approx \prod_{i=1}^{|E|} p(cmu_i|h_i) \tag{3}$$

where h_i denotes the history or context of e_i, which is described as follows:

$$h_i = \{e_i, e_{i+1}, e_{i+2}, e_{i-1}, e_{i-2}, cmu_{i-1}, cmu_{i-2}\} \quad . \tag{4}$$

History of an English phoneme is defined as its left-two and right-two neighboring phonemes plus the two *cmu*s at pinyin side, to which its left-two phonemes align. For each e in a given pair of $\{e_1, e_2, \ldots, e_n\}$ and $\{cmu_1, cmu_2, \ldots, cmu_n\}$, its conditional transliteration probability to produce the *cmu* with respect to its contextual history h can be computed by:

$$p(cmu|h) = \frac{p(h, cmu)}{\sum_{cmu' \in \Omega} p(h, cmu')} \tag{5}$$

where Ω is the set of all *cmu*s mapped from e and observed in the training data, and $p(h, cmu)$ is the joint probability distribution of observing h and *cmu*

Table 1. Baseline model feature templates

Category	Contextual Feature Templates	# of Possible Features
1	$e_i = \mathcal{X}$ and $cmu_i = \mathcal{Z}$	$\lvert\mathcal{V}_\mathcal{E}\rvert \cdot \lvert\mathcal{V}_\mathcal{C}\rvert$
2	$cmu_{i-1} = \mathcal{X}$ and $cmu_i = \mathcal{Z}$	$\lvert\mathcal{V}_\mathcal{C}\rvert^2$
3	$cmu_{i-2}cmu_{i-1} = \mathcal{X}\mathcal{Y}$ and $cmu_i = \mathcal{Z}$	$\lvert\mathcal{V}_\mathcal{C}\rvert^3$
4	$e_{i-1} = \mathcal{X}$ and $cmu_i = \mathcal{Z}$	$\lvert\mathcal{V}_\mathcal{E}\rvert \cdot \lvert\mathcal{V}_\mathcal{C}\rvert$
5	$e_{i-2} = \mathcal{X}$ and $cmu_i = \mathcal{Z}$	$\lvert\mathcal{V}_\mathcal{E}\rvert \cdot \lvert\mathcal{V}_\mathcal{C}\rvert$
6	$e_{i+1} = \mathcal{X}$ and $cmu_i = \mathcal{Z}$	$\lvert\mathcal{V}_\mathcal{E}\rvert \cdot \lvert\mathcal{V}_\mathcal{C}\rvert$
7	$e_{i+2} = \mathcal{X}$ and $cmu_i = \mathcal{Z}$	$\lvert\mathcal{V}_\mathcal{E}\rvert \cdot \lvert\mathcal{V}_\mathcal{C}\rvert$

simultaneously, which can be trained using maximum likelihood estimation. We use the *MaxEnt* model to solve the joint probability distribution [10]:

$$p(cmu, h) = \pi\mu \prod_{j=1}^{k} \alpha_j^{f_i(h,cmu)} \tag{6}$$

where π is a normalization constant, $\{\mu, \alpha_1, \alpha_2, \ldots, \alpha_k\}$ are the model parameters and $\{f_1, f_2, \ldots, f_k\}$ are features which are all binary. Each parameter α_j corresponds to a feature f_j. A feature takes the following form:

$$f_1(h_i, cmu_i) = \begin{cases} 1 & \text{if } e_i\text{=/F/ \& } e_{i-1}\text{=START \& } e_{i+1}\text{=/R/ \& } cmu_i\text{=/fu/} \\ 0 & \text{otherwise} \end{cases}$$

or

$$f_2(h_i, cmu_i) = \begin{cases} 1 & \text{if } e_i\text{=/S/ \& } e_{i+1}\text{=/T/ \& } cmu_{i-1}\text{=/IH/ \& } cmu_i\text{=/si/} \\ 0 & \text{otherwise .} \end{cases}$$

The general feature templates we used in experiments are listed in Table 1, where $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$ can be any individual English phoneme or Chinese pinyin *cmu*, and $|\mathcal{V}_\mathcal{E}|$ and $|\mathcal{V}_\mathcal{C}|$ are the number of elements in the respective sound vocabulary of English and Chinese.

4.2 Improving the Baseline Model

Deficiencies of the Baseline Model. There are two critical problems in the baseline model that can be improved:

1. The search space for finding the *Viterbi* alignment from all possible alignments is extremely large. Take the name pair in Fig. 3 for example, there is no means to prevent ill-formed *cmu*s, e.g. the first phoneme /F/ maps to /f/ and the second one /R/ maps to /ul/, which is an unfavorable alignment but cannot be avoided if such mappings dominate the training data. This could produce many illegal pinyin sequences and give rise to a large number of probable *cmu*s. They turned out adding uncertainties to phonetic transcriptions.

2. Because of compound pinyin finals, two consecutive English phonemes may map to a single pinyin symbol, such as mapping from /AE N/ to /ang/ in the example (see Fig. 2), which is not allowed in the baseline. This linguistic knowledge need not be imparted ad-hoc in the model. We can decompose compound finals into multiple basic finals, e.g. from /ang/ into /a/ and /ng/, to reduce the size of target phonetic vocabulary. The original mapping is broken into /AE/-to-/a/ and /N/-to-/ng/.

Precise Alignment of Phoneme Chunks. We introduce alignment indicators between a pair of sound sequences of E and C. Within 39 English phonemes (24 consonants, 15 vowels) and 58 pinyin symbols (23 initials and 35 finals), there are always some indicative elements for alignment, i.e. indicators. For E, they are all the consonants, the vowel at the first position and the second vowel of two contiguous vowels; for C correspondingly, they are all the initials, the final at the first position and the second final of two contiguous finals. Also, we define the following variables: $\tau(S)$ is defined as # of indicators in sequence S ($S \in \{E, C\}$); $t(E, C) = \max\{\tau(E), \tau(C)\}$ represents the maximum # of indicators in E and C; $d(E, C) = |\tau(E) - \tau(C)|$ is the difference of the # of indicators in E and C.

We chunk E and C by tagging the identified indicators and compensate the one with fewer indicators by inserting d number of mute ε at its min $\{\tau(E), \tau(C)\}$ possible positions ahead of its indicators. ε is practically an indicator defined for alignment. This ensures that both sequences end up with the same number of indicators. The t chunks separated by indicators in E should align to the corresponding t chunks in C in the same order. They are called alignment chunks. There are $\|A\| = \binom{d}{t} = \frac{t!}{(t-d)!d!}$ number of possible alignments at chunk level with respect to different positions of ε.

This method can guarantee each chunk contains two sound units at most. Thus, in a pair of aligned chunks, only three mapping layouts between phoneme elements are possible:

1. e-to-c_1c_2: The alignment would be e-to-c_1c_2 where c_1c_2 is considered as an initial-final cluster (*cmu*);
2. e_1e_2-to-$c1c_2$: The alignment at phoneme level would be extended to e_1-to-c_1 and e_2-to-c_2. Note that no new alignment is generated under this condition. Thus, the total number of alignments remains unchanged;
3. e_1e_2-to-c: By adding an additional ε at C side, the alignment at phoneme level would be extended to e_1-to-c and e_2-to-ε or e_1-to-ε and e_2-to-c. In this case, one more new alignment will be produced and we update $\|A\| = \|A\|+1$.

EM **Training for Symbol-Mappings.** We then applied *EM* algorithm [4, 7] to find the *Viterbi* alignment for each training pair as follows:

1. *Initialization:* For each English-Chinese pair, assign equal weights to all alignments generated based on phoneme chunks as $\|A\|^{-1}$.

2. *Expectation Step:* For each of the 39 English phonemes, count the instances of its different mappings from the observations on all alignments produced. Each alignment contributes counts in proportion to its own weight. Normalize the scores of the mapping units it maps to so that the mapping probability sums to 1.
3. *Maximization Step:* Re-compute the alignment scores. Each alignment is scored with the product of the scores of the symbol mappings it contains. Normalize the alignment scores so that each pair's scores sum to 1.
4. *Repeat* step 2-3 until the symbol-mapping probabilities converge, meaning that the variation of each probability between two iterations becomes less than a specified threshold.

With the improved alignment, the mappings crossing chunks are avoided. Thus, the *EM* training becomes more precise and produces significantly fewer possible alignments compared to the baseline.

5 Experiments and Evaluation

5.1 Data Set

We obtained the beta release v.1.0 of LDC's Chinese-English bi-directional named entity list compiled from Xinhua's database, from which we chose the English-to-Chinese proper name list of people as raw data. The list contains 572,213 foreign people's names and their Chinese transliterations. Note that although the list is in English, it contains names originated from different languages (e.g. Russian, German, Spanish, Arabic, Japanese, Korean, etc.). One assumption is that the Chinese translations were produced based on their English pronunciations directly. The exceptions are Japanese and Korean names, which are generally translated in terms of meaning as opposed to pronunciation, and we consider them as noise. We resorted to CMU's pronunciation dictionary and LDC's Chinese character table with pinyin to convert the names into a parallel corpus of sequences of English phonemes and pinyin symbols. We ended up with 46,305 pairs, which were then used as our experimental data pool.

5.2 Performance Measurement

There is no standard for measuring machine transliteration. Some tests require human judgment. The performance was evaluated with two levels of accuracy, i.e. character-level accuracy ($C.A.$) and word level accuracy ($W.A.$) in [6]:

$$C.A. = \frac{L - (i + d + s)}{L} \tag{7}$$

$$W.A. = \frac{\#\ of\ correct\ names\ generated}{\#\ of\ tested\ names}\ . \tag{8}$$

In (7), L is the length of the standard transliteration of a given foreign name, and i, d, and s are the number of insertion, deletion and substitution respectively,

i.e. edit distance between machine-generated transliteration and the standard. If $L < (i+d+s)$, we set $C.A. = 0$. Equation (8) is the percentage of the number of transliterations identical to the standards in all the tested names. A name often has acceptable transliteration alternatives. Hence, we will also measure how the percentage of the number of transliterations distributes over different character-level accuracy ranges, which is referred to as *C.A.* Distribution (*C.A.D.*):

$$C.A.D. = \frac{\#\ of\ names\ with\ C.A. \in [r_1, r_2)}{\#\ of\ tested\ names} \tag{9}$$

where $[r_1, r_2)$ (denotes $r_1 \leq C.A. < r_2$) is the bound of a *C.A.* range. We set up six *C.A.* ranges: [0%, 20%), [20%, 40%), [40%, 60%), [60%, 80%), [80%, 100%) and [100%]. We are especially interested in the names within the ranges [0%, 20%) and [80%, 100%) since the former could be considered as "completely incorrect" and the latter "acceptable".

5.3 Experiments and Results

In each trial of our experiments, individual translation name pairs, hereafter referred to as instances, were randomly selected from the data pool to build 10 subsets. Each respectively accounts for 10% to 100% (step=10%) of the total instances in the entire pool. In each subset, we used 90% of the instances for training and the remaining 10% for open test. Also the same number of instances (10%) were randomly selected from the training data for close test.

Experiments. The baseline model was trained and tested as follows:

1. Using *EM* iterations in GIZA++ to obtain *Viterbi* alignment of each pair of names in the training set. The bootstrapping settings were the same as [13] (see Sect. 2). Note that the direction of estimation is from E to C directly;
2. Aligned training instances were then passed to *GIS* (Generalized Iterative Scaling) algorithm for training the *MaxEnt* models [1, 10]. This fulfilled training the models that can transliterate phoneme sequences of given English names into pinyin sequences;
3. Tests were conducted on the trained *MaxEnt* models. "Beam search" [10] was used where a beam size of 5 was adopted. For each given name, only top-1 transliteration was accepted.

The experiments on the improved model were conducted under similar settings except that the tailored alignment scheme and the *EM* training (see Sect. 4.2) were applied in the step 1.

To investigate the influence of data sizes on performance, the above procedure was applied to the 10 subsets with different data sizes as described previously. And the performance of the model was measured by the average accuracy (*C.A.* and *W.A.*) of 50 trials of the experiments. *C.A.D.* was measured with average distributions of 50 trials on 100% data size only.

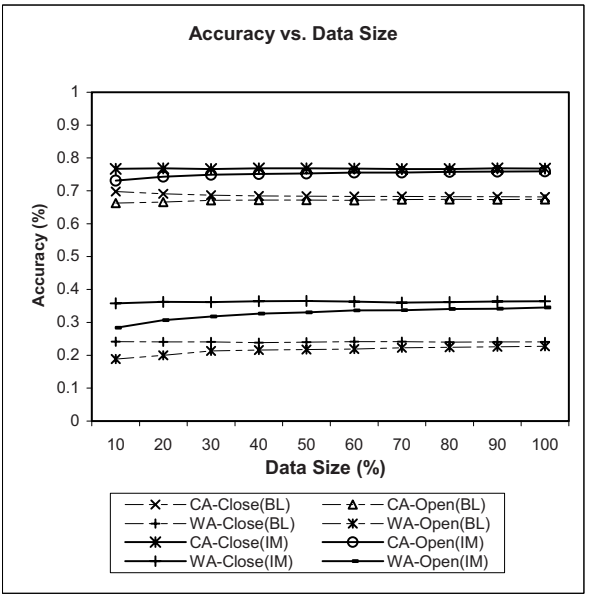

Fig. 4. Comparison of baseline (BL) and improved (IM) models on *C.A.*&*W.A.* vs. Data Size

Results and Discussions. Figure 4 shows the average *C.A.* and *W.A.* of the baseline model (BL) and the improved model (IM) over different data sizes. IM significantly outperforms BL on all tests. In open tests with 100% data, for example, IM demonstrates improvements on *C.A.* by 8.56% and on *W.A.* by 11.84%. Recall that in the IM model, we chopped longer pinyin symbols, e.g. compound finals, into smaller sound units, i.e. basic finals, and aligned chunks of English phonemes with corresponding chunks of pinyin symbols, prohibiting alignments across chunk borders. This could produce: 1. more precise mappings between English phonemes and *cmu*s; 2. less possible *cmu*s for each English phoneme, reducing uncertainties; 3. less *cum*s forming illegal pinyin syllables, leading to more legitimate pinyin sequences. The figure also shows that with enough instances, the models could achieve almost equal performance in open tests to closed ones.

Figure 5 shows the average percentage of the number of transliterations distributed over their *C.A.* values (on all data). For *C.A.* ranging from 0% to 80%, BL produced more transliterations throughout the four ranges than IM. In the remaining *C.A.* ranges, IM produced more high-quality transliterations (see $C.A. \geq 80\%$) and considerably more correct transliterations (see $C.A. = 100\%$) than BL.

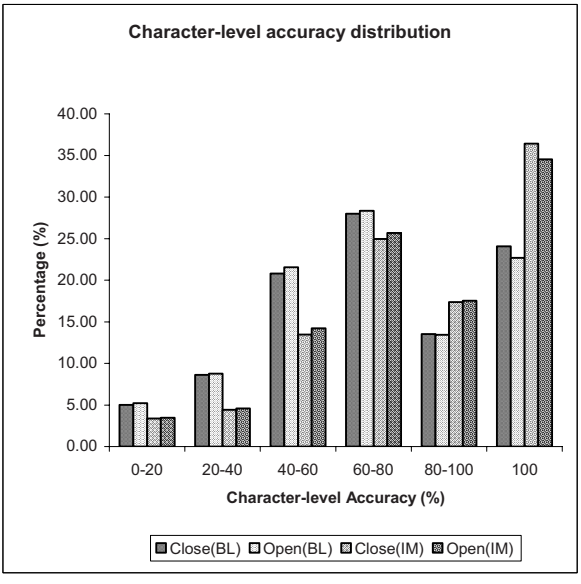

Fig. 5. Comparison of baseline (BL) and improved (ML) models on *C.A.D.*

Table 2. Results of comparisons on different systems

Systems/Accuracy		SC	BL	IM
C.A.	Close	66.35%	68.18%	76.97%
	Open	65.15%	67.18%	75.08%
W.A.	Close	20.73%	23.47%	36.19%
	Open	18.27%	21.49%	32.50%

5.4 Comparisons with Source-Channel System

We compared our work with the source-channel (SC) system described in [13]. Their method (the first translation system) was replicated with the only exception that we obtained phoneme sequences of foreign names via a lookup of CMU's pronunciation dictionary, whereas they adopted the Festival text-to-speech system for English pronunciations. Then we tested the SC system, our BL system and IM system using the entire data pool with 41,674 instances for training and the remaining 4,631 for testing. The language model of SC system was trained on the 41,674 pinyin sequences in the training portion, similarly using trigram by CMU-Cambridge toolkits as [13]. The results are shown in Table 2. Our BL and IM system outperformed the source-channel approach by about 3% and 10% respectively in all tests using the same data set.

6 Conclusion

We modeled English-to-Chinese transliteration as direct phonetic mapping from English phonemes to a set of basic pinyin symbols plus dynamically discovered mapping units from training. Contextual features of each phoneme are taken into consideration in the model. An effective algorithm for precise alignment of phoneme chunks was presented, which demonstrated improvements on performance. Comparisons show that our approaches significantly outperforms traditional source-channel model. Future work will include incorporating different features, such as additional contexts, target language model, or even the composition of direct and inverted transliteration model under *MaxEnt* framework.

Acknowledgement

The work described in this paper is partially supported by CUHK under the Strategic Grant initiative (Project Account No.: 4410001). We are especially grateful to Prof. Helen Meng for her support and help on obtaining LDC corpora.

References

1. Berger, A.L., Della Pietra, S.A., and Della Pietra, V.J.: A Maximum entropy approach to natural language processing. Computational Linguistics **22** (1996) 39–27
2. Brown, P.F., Della Pietra, S.A., Della Pietra, V.J., and Mercer, R.L.: The mathematics of statistical machine translation: Parameter Estimation. Computational Linguistics **19** (1993) 261–311
3. Clarkson, P., and Ronsenfeld, R.: Statistical language modeling using the CMU-Cambridge toolkit. In Proc. of the 5th EuroSpeech (1997) 2707–2710
4. Gao, W., Wong, K.F., and Lam, W.: Phoneme-based transliteration of foreign names for OOV problem. In Proc. of IJCNLP (2004) 374–381
5. Germann, U., Jahr, M., Knight, K., Marcu, D., and Yamada, K.: Fast decoding and optimal decoding for machine translation. In Proc. of ACL (2001) 228–235
6. Kang, I.H., and Kim, C.C.: English-to-Korean transliteration using multiple unbounded overlapping phoneme chunks. In Proc. of COLING (2000) 418–424
7. Knight, K., and Graehl, J.: Machine transliteration. In Proc. of ACL (1997) 128–135
8. Lee, J.S., and Choi, K.S.: English to Korean statistical transliteration for information retrieval. Computer Processing of Oriental Languages **12** (1998) 17–27
9. Och, F.J., and Ney, H.: Discriminative training and maximum entropy models for statistical machine translation. In Proc. of ACL (2002) 295–302
10. Ratnaparkhi, A.: A maximum entropy model for Part-Of-Speech tagging. In Proc. of EMNLP (1996) 133–141
11. Rabiner, L.R.: A tutorial on Hidden Markov Models and selected applications in speech recognition. In Proc. of IEEE **77** (1989) 257–286
12. Stalls, B.G., and Knight, K.: Translating names and technical terms in Arabic text. In Proc. of COLING/ACL Workshop on Computational Approaches to Semitic Languages (1998)
13. Virga, P., and Khudanpur, S.: Transliteration of proper names in cross-lingual information retrieval. In Proc. of ACL Workshop on Multi-lingual Named Entity Recognition (2003)