A Log-linear Block Transliteration Model based on Bi-Stream HMMs

Bing Zhao, Nguyen Bach, Ian Lane, and **Stephan Vogel**
{bzhao, nbach, ianlane, vogel}@cs.cmu.edu
Language Technologies Institute
School of Computer Science, Carnegie Mellon University

Abstract

We propose a novel HMM-based framework to accurately transliterate unseen named entities. The framework leverages features in letter-alignment and letter n-gram pairs learned from available bilingual dictionaries. Letter-classes, such as vowels/non-vowels, are integrated to further improve transliteration accuracy. The proposed transliteration system is applied to out-of-vocabulary named-entities in statistical machine translation (SMT), and a significant improvement over traditional transliteration approach is obtained. Furthermore, by incorporating an automatic spell-checker based on statistics collected from web search engines, transliteration accuracy is further improved. The proposed system is implemented within our SMT system and applied to a real translation scenario from Arabic to English.

1 Introduction

Cross-lingual natural language applications, such as information retrieval, question answering, and machine translation for web-documents (e.g. Google translation), are becoming increasingly important. However, current state-of-the-art statistical machine translation (SMT) systems cannot yet translate named-entities which are not seen during training. New named-entities, such as person, organization, and location names are continually emerging on the World-Wide-Web. To realize effective cross-lingual natural language applications, handling out-of-vocabulary named-entities is becoming more crucial.

Named entities (NEs) can be translated via transliteration: mapping symbols from one writing system to another. Letters of the source language are typically transformed into the target language with similar pronunciation. Transliteration between languages which share similar alphabets and sound systems is usually not difficult, because the majority of letters remain the same. However, the task is significantly more difficult when the language pairs are considerably different, for example, English-Arabic, English-Chinese, and English-Japanese. In this paper, we focus on *forward* transliteration from Arabic to English.

The work in (Arbabi et al., 1994), to our knowledge, is the first work on machine transliteration of Arabic names into English, French, and Spanish. The idea is to vowelize Arabic names by adding appropriate vowels and utilizing a phonetic look-up table to provide the spelling in the target language. Their framework is strictly applicable within standard Arabic morphological rules. Knight and Graehl (1997) introduced finite state transducers that implement back-transliteration from Japanese to English, which was then extended to Arabic-English in (Stalls and Knight, 1998). Al-Onaizan and Knight (2002) transliterated named entities in Arabic text to English by combining phonetic-based and spelling-based models, and reranking candidates with full-name web counts, named entities co-reference, and contextual web counts. Huang (2005) proposed a specific model for Chinese-English name transliteration with clusterings of names' origins, and appropriate hypotheses are generated given the origins. All of these approaches, however, are not based on a SMT-framework. Technologies developed for SMT are borrowed in Virga and Khudanpur (2003) and AbdulJaleel and Larkey (2003). Standard SMT alignment models (Brown et al., 1993) are used to align letter-pairs within named entity pairs for transliteration. Their approach are generative models for letter-to-letter translations, and the letter-alignment is augmented with heuristics. Letter-level contextual information is shown to be very helpful for transliteration. Oh and Choi (2002) used conversion units for English-Korean Transliteration; Goto et al. (2003) used conversion units, mapping English letter-sequence into Japanese Katakana character string. Li et al. (2004) presented a framework allowing direct orthographical mapping of transliteration units between English and Chinese, and an extended model is presented in Ekbal et al. (2006).

We propose a *block-level* transliteration framework, as shown in Figure 1, to model letter-level context information for transliteration at two levels. First, we propose a bi-stream HMM incorporating letter-clusters to better model the *vowel* and *non-vowel* transliterations with position-information, i.e., *initial* and *final*, to improve the letter-level alignment accuracy. Second, based on the letter-alignment, we propose *letter n-gram* (letter-sequence) alignment models (*block*) to automatically learn the mappings from source letter n-grams to target letter n-grams. A few features specific for transliterations are explored, and a log-linear model is used to combine

Proceedings of NAACL HLT 2007, pages 364–371,
Rochester, NY, April 2007. ©2007 Association for Computational Linguistics

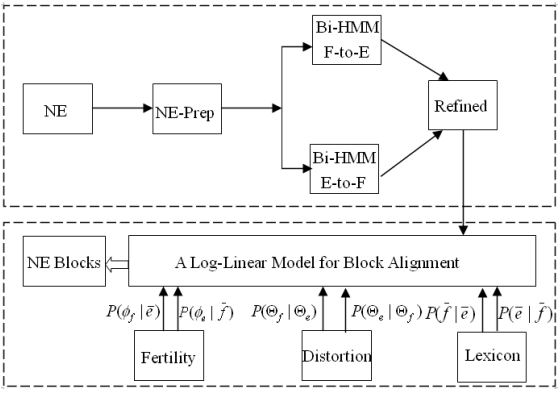

Figure 1: Transliteration System Structure. The upper-part is the two-directional Bi-Stream HMM for letter-alignment; the lower-part is a log-linear model for combining different feature functions for block-level transliteration.

these features to learn block-level transliteration-pairs from training data. The proposed transliteration framework obtained significant improvements over a strong baseline transliteration approach similar to AbdulJaleel and Larkey (2003) and Virga and Khudanpur (2003).

The remainder of this paper is organized as follows. In Section 2, we formulate the transliteration as a general translation problem; in Section 4, we propose a log-linear alignment model with a local search algorithm to model the letter n-gram translation pairs; in Section 5, experiments are presented. Conclusions and discussions are given in Section 6.

2 Transliteration as Translation

Transliteration can be viewed as a special case of translation. In this approach, source and target NEs are split into letter sequences, and each sequence is treated as a *pseudo sentence*. The appealing reason of formulating transliteration in this way is to utilize advanced alignment models, which share ideas applied also within phrase-based statistical machine translation (Koehn, 2004).

To apply this approach to transliteration, however, some unique aspects should be considered. First, letters should be generated from left to right, without any reordering. Thus, the transliteration models can only execute forward sequential jumps. Second, for unvowelized languages such as Arabic, a single Arabic letter typically maps to less than four English letters. Thus, the fertility for each letter should be recognized to ensure reasonable length relevance. Third, the position of the letter within a NE is important. For example, in Arabic, letters such as "al" at the beginning of the NE can only be translated into "the" or "al". Therefore position information should be considered within the alignment models.

Incorporating the above considerations, transliteration can be formulated as a noisy channel model. Let $f_1^J = f_1 f_2 ... f_J$ denote the source NE with J letters, $e_1^I = e_1 e_2 ... e_I$ be an English transliteration candidate with I letters. According to Bayesian decision rule:

$$\hat{e}_1^I = \arg\max_{\{e_1^I\}} P(e_1^I|f_1^J) = \arg\max_{\{e_1^I\}} P(f_1^J|e_1^I)P(e_1^I), \quad (1)$$

where $P(f_1^J|e_1^I)$ is the *letter translation model* and $P(e_1^I)$ is the English *letter sequence model* corresponding to the monolingual language models in SMT. In this noisy-channel scheme, $P(f_1^J|e_1^I)$ is the key component for transliteration, in which the transliteration between e_1^I and f_1^J can be modeled at either letter-to-letter level, or letter n-gram transliteration level (*block*-level).

Our transliteration models are illustrated in Figure 1. We propose a Bi-Stream HMM of $P(f_1^J|e_1^I)$ to infer letter-to-letter alignments in two directions: Arabic-to-English (F-to-E) and English-to-Arabic (E-to-F), shown in the upper-part in Figure 1; refined alignment is then obtained. We propose a log-linear model to extract block-level transliterations with additional informative features, as illustrated in the lower-part of Figure 1.

3 Bi-Stream HMMs for Transliteration

Standard IBM translation models (Brown et al., 1993) can be used to obtain letter-to-letter translations. However, these models are not directly suitable, because letter-alignment within NEs is strictly left-to-right. This sequential property is well suited to HMMs (Vogel et al., 1996), in which the jumps from the current aligned position can only be forward.

3.1 Bi-Stream HMMs

We propose a bi-stream HMM for letter-alignment within NE pairs. For the source NE f_1^J and a target NE e_1^I, a bi-stream HMM is defined as follows:

$$p(f_1^J|e_1^I) = \sum_{a_1^J} \prod_{j=1}^{J} p(f_j|e_{a_j}) p(c_{f_j}|c_{e_{a_j}}) p(a_j|a_{j-1}), \quad (2)$$

where a_j maps f_j to the English letter e_{a_j} at the position a_j in the English named entity. $p(a_j|a_{j-1})$ is the transition probability distribution assuming first-order Markov dependency; $p(f_j|e_{a_j})$ is a letter-to-letter translation lexicon; c_{f_j} is the letter cluster of f_j and $p(c_{f_j}|c_{e_{a_j}})$ is a cluster level translation lexicon. As mentioned in the above, the vowel/non-vowel linguistic features can be utilized to cluster the letters. The letters from the same cluster tend to share the similar letter transliteration forms. $p(c_{f_j}|c_{e_{a_j}})$ enables to leverage such letter-correlation in the transliteration process.

The HMM in Eqn. 2 generates two streams of observations: the letters together with the letters' classes following the distribution of $p(f_j|e_{a_j})$ and $p(c_{f_j}|c_{e_{a_j}})$ at each

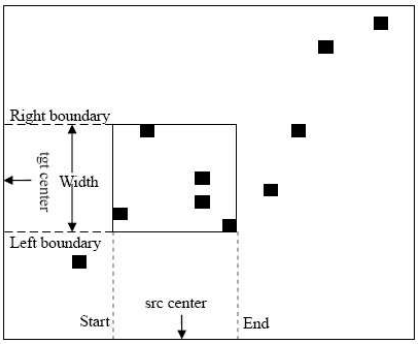

Figure 2: Block of letters for transliteration. A block is defined by the left- and right- boundaries in the NE-pair.

state, respectively. To be in accordance with the monotone nature of the NE's alignment mentioned before, we enforce the following constraints in Eqn. 3, so that the transition can only jump forward or stay at the same state:

$$a_j - a_{j-1} \geq 0 \qquad \forall j \in [1, J]. \tag{3}$$

Since the two streams are conditionally independent given the current state, the extended EM is straightforward, with only small modifications of the standard forward-backward algorithm (Zhao et al., 2005), for parameter estimation.

3.2 Designing Letter-Classes

Pronunciation is typically highly structured. For instance, in English the pronunciation structure of "*cvc*" (*consonant-vowel-consonant*) is common. By incorporating letter classes into the proposed two-stream HMM, the models' expressiveness and robustness can be improved. In this work, we focus on transliteration of Arabic NEs into English. We define six non-overlapping letter classes: *vowel*, *consonant*, *initial*, *final*, *noclass*, and *unknown*. *Initial* and *final* classes represent semantic markers at the beginning or end of NEs such as "Al" and "wAl" (in romanization form). *Noclass* signifies letters which can be pronounced as both a vowel and a consonant depending on context, for example, the English letter "y". The *unknown* class is reserved for punctuations and letters that we do not have enough linguistic clues for mapping them to phonemes.

4 Transliteration Blocks

To further leverage the information from the letter-context beyond the letter-classes incorporated in our bi-stream HMM in Eqn. 2, we define *letter n-grams*, which consist of n *consecutive letters*, as the basic transliteration unit. A *block* is defined as a pair of such letter n-grams which are transliterations of each other. During decoding of unseen NEs, transliteration is performed block-by-block, rather than letter-by-letter. The goal of transliteration model is to learn high-quality transliteration blocks from the training data in a unsupervised fashion.

Specifically, a block X can be represented by its left and right boundaries in the source and target NEs shown in Figure 2:

$$X = (f_j^{j+l}, e_i^{i+k}), \tag{4}$$

where f_j^{j+l} is the source letter-ngram with $(l+1)$ letters in source language, and its projection of e_i^{i+k} in the English NE with left boundary at the position of i, and right boundary at $(i+k)$.

We formulate the *block extraction* as a *local search* problem following the work in Zhao and Waibel (2005): given a source letter n-gram f_j^{j+l}, search for the projected boundaries of candidate target letter n-gram e_i^{i+k} according to a weighted combination of the diverse features in a *log-linear model* detailed in §4.3. The log-linear model serves as a performance measure to guide the local search, which, in our setup, is *randomized hill-climbing*, to extract bilingual letter n-gram transliteration pairs.

4.1 Features for Block Transliteration

Three features: *fertility*, *distortion*, and *lexical translation* are investigated for inferring transliteration blocks from the NE pairs. Each feature corresponds to one aspect of the block within the context of a given NE pair.

4.1.1 Letter n-gram Fertility

The *fertility* $P(\phi|e)$ of a target letter e specifies the probability of generating ϕ source letters for transliteration. The fertilities can be easily read-off from the letter-alignment, i.e., the output from the Bi-stream HMM. Given letter fertility model $P(\phi|e_i)$, a target letter n-gram e_1^I, and a source n-gram f_1^J of length J, we compute a probability of *letter n-gram length* relevance: $P(J|e_1^I)$ via a dynamic programming.

The probability of generating J letters by the English letter n-gram e_1^I is defined:

$$P(J|e_1^I) = \max_{\{\phi_1^I, J=\sum_{i=1}^I \phi_i\}} \prod_{i=1}^{I} P(\phi_i|e_i). \tag{5}$$

The recursively updated cost $\phi[j,i]$ in dynamic programming is defined as follows:

$$\phi[j,i] = max \begin{cases} \phi[j,i-1] + \log P_{Null}(0|e_i) \\ \phi[j-1,i-1] + \log P_\phi(1|e_i) \\ \phi[j-2,i-1] + \log P_\phi(2|e_i) \\ \phi[j-3,i-1] + \log P_\phi(3|e_i) \end{cases}, \tag{6}$$

where $P_{Null}(0|e_i)$ is the probability of generating a Null letter from e_i; $P_\phi(k{=}1|e_i)$ is the letter-fertility model of generating *one* source letter from e_i; $\phi[j,i]$ is the cost

so far for generating j letters from i consecutive English letters (letter n-gram) $e_1^i : e_1, \cdots, e_i$.

After computing the cost of $\phi[J, I]$, the probability $P(J|e_1^I)$ is computed for generating the length of the source NE f_1^J from the English NE e_1^I shown in Eqn. 5. With this letter n-gram fertility model, for every block, we can compute a fertility score to estimate how relevant the lengths of the transliteration-pairs are.

4.1.2 Distortion of Centers

When aligning blocks of letters within transliteration pairs, we expect most of them are close to the diagonal due to the monotone alignment nature. Thus, a simple position metric is proposed for each block considering the relative positions within NE-pairs.

The center $\odot_{f_j^{j+l}}$ of the source phrase f_j^{j+l} with a length of $(l+1)$ is simply a normalized relative position in the source entity defined as follows:

$$\odot_{f_j^{j+l}} = \frac{1}{l+1} \sum_{j'=j}^{j'=j+l} \frac{j'}{l+1}. \quad (7)$$

For the center of English letter-phrase e_i^{i+k}, we first define the expected corresponding relative center for every source letter $f_{j'}$ using the lexicalized position score as follows:

$$\odot_{e_i^{i+k}}(f_{j'}) = \frac{1}{k+1} \cdot \frac{\sum_{i'=i}^{(i+k)} i' \cdot P(f_{j'}|e_{i'})}{\sum_{i'=i}^{(i+k)} P(f_{j'}|e_{i'})}, \quad (8)$$

where $P(f_{j'}|e_i)$ is the letter translation lexicon estimated in IBM Models $1{\sim}5$. i is the position index, which is weighted by the letter-level translation probabilities; the term of $\sum_{i'=i}^{i+k} P(f_{j'}|e_{i'})$ provides a normalization so that the expected center is within the range of the target length. The expected center for e_i^{i+k} is simply the average of the $\odot_{e_i^{i+k}}(f_{j'})$:

$$\odot_{e_i^{i+k}} = \frac{1}{l+1} \sum_{j'=j}^{j+l} \odot_{e_i^{i+k}}(f_{j'}) \quad (9)$$

Given the estimated centers of $\odot_{f_j^{j+l}}$ and $\odot_{e_i^{i+k}}$, we can compute how close they are via the probability of $P(\odot_{f_j^{j+l}}|\odot_{e_i^{i+k}})$. In our case, because of the monotone alignment nature of transliteration pairs, a simple gaussian model is employed to enforce that the point $(\odot_{e_i^{i+k}}, \odot_{f_j^{j+l}})$ is not far away from the diagonal.

4.1.3 Letter Lexical Transliteration

Similar to IBM Model-1 (Brown et al., 1993), we use a "bag-of-letter" generative model within a block to approximate the lexical transliteration equivalence:

$$P(f_j^{j+l}|e_i^{i+k}) = \prod_{j'=j}^{j+l} \sum_{i'=i}^{i+k} P(f_{j'}|e_{i'}) P(e_{i'}|e_i^{i+k}), \quad (10)$$

where $P(e_{i'}|e_i^{i+k}) \simeq 1/(k+1)$ is approximated by a bag-of-word unigram. Since named entities are usually relatively short, this approximation works reasonably well in practice.

4.2 Extended Feature Functions

Because of the underlying nature of the noisy-channel model in our proposed transliteration approach in Section 2, the three base feature functions are extended to cover the directions both from target-to-source and source-to-target. Therefore, we have in total six feature functions for inferring transliteration blocks from a named entity pair.

Besides the above six feature functions, we also compute the average letter-alignment links per block. We count the number of letter-alignment links within the block, and normalize the number by the length of the source letter-ngram. Note that, we can refine the letter-alignment by growing the intersections of the two direction letter-alignments from Bi-stream HMM via additional aligned letter-pairs seen in the union of the two. In a way, this approach is similar to those of refining the word-level alignment for SMT in (Och and Ney, 2003). This step is shown in the upper-part in Figure 1.

Overall, our proposed feature functions cover relatively different aspects for transliteration blocks: the block level length relevance probability in Eqn. 5, lexical translation equivalence, and positions' distortion from a gaussian distribution in Eqn. 8, in both directions; and the average number of letter-alignment links within the block. Also, these feature functions are positive and bounded within $[0, 1]$. Therefore, it is suitable to apply a log-linear model (in §4.3) to combine the *weighted* individual strengths from the proposed feature functions for better modeling the quality of the candidate transliteration blocks. This log-linear model will serve as a performance measure in a local-search in §4.4 for inferring transliteration blocks.

4.3 Log-Linear Transliteration Model

We propose a log-linear model to combine the seven feature functions in §4.1 with proper weights as in Eqn. 11:

$$Pr(X|\mathbf{e}, \mathbf{f}) = \frac{\exp(\sum_{m=1}^{M} \lambda_m \phi_m(X, \mathbf{e}, \mathbf{f}))}{\sum_{\{X'\}} \exp(\sum_{m=1}^{M} \lambda_m \phi_m(X', \mathbf{e}, \mathbf{f}))}, \quad (11)$$

where $\phi_m(X, \mathbf{e}, \mathbf{f})$ are the real-valued bounded feature functions corresponding to the seven models introduced in §4.1. The log-linear model's parameters are the weights $\{\lambda_m\}$ associated with each feature function.

With hand-labeled data, $\{\lambda_m\}$ can be learnt via generalized iterative scaling algorithm (GIS) (Darroch and Ratcliff, 1972) or improved iterative scaling (IIS) (Berger

et al., 1996). However, as these algorithms are computationally expensive, we apply an alternative approach using a simplex down-hill algorithm to optimize the weights toward better F-measure of block transliterations. Each feature function corresponds to one dimension in the simplex, and the local optimum only happens at a vertex of the simplex. Simplex-downhill has several advantages: it is an efficient approach for optimizing multi-variables given some performance measure. We compute the F-measure against a gold-standard block set extracted from hand-labeled letter-alignment.

To build gold-standard blocks from hand-labeled letter-alignment, we propose the *block transliteration coherence* in a two-stage fashion. First is the forward projection: for each candidate source letter-ngram f_j^{j+n}, search for its *left-most* e_l and *right-most* e_r projected positions in the *target* NE according to the given letter-alignment. Second is the backward projection: for the target letter-gram e_l^r, search for its *left-most* $f_{l'}$ and *right-most* $f_{r'}$ projected positions in the *source* NE. Now if $l' \geq j$ and $r' \leq j+n$, i.e. f_l^r is contained within the source letter-ngram f_j^{j+n}, then this block $X = (f_j^{j+n}, e_l^r)$ is defined as *coherent* for the aligned pairs: (f_j^{j+n}, e_l^r) . We accept coherent X as gold-standard blocks. This block transliteration coherence is generally sound for extracting the gold-blocks mostly because of the the monotone left-to-right nature of the letter-alignment for transliteration. A related coherence assumption can be found in (Fox, 2002), where their assumption on phrase-pairs for statistical machine translation is shown to be somewhat restrictive for SMT. This is mainly because the word alignment is often *non-monotone*, especially for langauge-pairs from different families such as Arabic-English and Chinese-English.

4.4 Aligning Letter-Blocks: a Local Search

Aligning the blocks within NE pairs can be formulated as a local search given the heuristic function defined in Eqn. 11. To be more specific: given a Arabic letter-ngram f_j^{j+l}, our algorithm searches for the best translation candidate e_i^{i+k} in the target named entities. In our implementation, we use stochastic hill-climbing with Eqn. 11 as the performance measure. Down-hill moves are accepted to allow one or two left and right null letters to be attached to e_i^{i+k} to expand the table of transliteration-blocks.

To make the local search more effective, we normalize the letter translation lexicon $p(f|e)$ within the parallel entity pair as in:

$$\hat{P}(f|e) = \frac{P(f|e)}{\sum_{j'=1}^{J} P(f_{j'}|e)}. \qquad (12)$$

In this way, the distribution of $\hat{P}(f|e)$ is sharper and more focused in the context of an entity pair.

Overall, given the parallel NE pairs, we can train the letter level translation models in both directions via the Bi-stream HMM in Eqn. 2. From the letter-alignment, we can build the letter translation lexicons and fertility tables. With these tables, the base feature functions are then computed for each candidate block, and the features are combined in the log-linear model in Eqn. 11. Given a named-entity pair in the training data, we rank all the transliteration blocks by the scores using the log-linear model. This step is shown in the lower-part in Figure 1.

4.5 Decoding Unseen NEs

The decoding of NEs is an extension to the noisy-channel scheme in Eqn. 1. In our configurations for NE transliteration, the extracted transliteration blocks are used. Our letter ngram is a standard letter-ngram model trained using the SriLM toolkit (Stolcke, 2002). To transliterate the unseen NEs, the decoder (Hewavitharana et al., 2005) is configured for monotone decoding. It loads the transliteration blocks and the letter-ngram LM, and it decodes the unseen Arabic named entities with block-based transliteration from left to right.

5 Experiments

5.1 The Data

We have 74,887 bilingual geographic names from LDC2005G01-NGA, 11,212 bilingual person names from LDC2005G02[1], and about 6,000 bilingual names extracted from the BAMA[2] dictionary. In total, there are 92,099 NE pairs. We split them into three parts: 91,459 pairs as the training dataset, 100 pairs as the development dataset, and 540 unique NE pairs as the held-out dataset.

An additional test set is collected from the TIDES 2003 Arabic-English machine translation evaluation test set. The 663 sentences contain 286 unique words, which were not covered by the available training data. From this set of untranslated words, we manually labeled the entities of persons, locations and organizations, giving a total of 97 unique un-translated NEs. The BAMA toolkit was used to romanize the Arabic words. Some names from this test set are shown in Figure 1.

These untranslated NEs make up only a very small fraction of all words in the test set. Therefore, having correct transliterations would give only small improvements in terms of BLEU (Papineni et al., 2002) and NIST scores. However, successfully translating these unknown NEs is very crucial for cross-lingual distillation tasks or question-answering based on the MT-output.

[1]The corpus is provided as FOUO (for official use only) in the *DARPA-GALE* project

[2]LDC2004L02: Buckwalter Arabic Morphological Analyzer version 2.0

Table 1: Test Set Examples.

Arabic	BAMA	Reference
غرابو	grAbw	Grabo
قشطة	qXTp	Qishta
ايتساخرف	fAytsAxr	Weizsacker
والدماني	wAldHmAny	al-Dahmani
يلويغيرز	zylwygyr	Zellweger
ثاكسين	vAksyn	Thaksin

To evaluate the transliteration performance, we use *edit-distance* between the hypothesis against a reference set. This is to count the number of insertions, deletions, and substitutions required to correct the hypothesis to match the given reference. An edit-distance of zero is a perfect match. However, NEs typically have more than one correct variant. For example, the Arabic name "mHmd" (in romanized form) can be transliterated as Muhammad or Mohammed; both are considered as correct transliterations. Ideally, we want to have all variants as reference transliterations. To enable our transliteration evaluation to be more informative given only one reference, edit-distance of one between hypothesis and reference is considered to be an acceptable match.

5.2 Comparison of Transliteration Models

We compare the performance of three systems within our proposed framework in Figure.1: the baseline Block system, a system in which we use a log-linear combination of alignment features as described in §4.3, we call the the L-Block system, and finally a system, which also uses the bi-stream HMM alignment model as described in §3. This last system will be denoted LCBE system.

The baseline is based on the refined letter-alignment from the two directions of IBM-Model-4, trained with a scheme of $1^5h^54^5$ using GIZA++ (Och and Ney, 2004). The final alignment was obtained by growing the intersections between Arabic-to-English (AE) and English-to-Arabic (EA) alignments with additional aligned letter-pairs seen in the union. This is to compensate for the inherent asymmetry in alignment models. Blocks (letter-ngram pairs) were collected directly from the refined letter-alignment, using the same algorithm as described in §4.3 for extracting gold-standard letter blocks. There is no length restrictions to the letter-ngram extracted in our system. All the blocks were then scored using relative frequencies and lexical scores in both directions, similar to the scoring of phrase-pairs in SMT (Koehn, 2004).

In the L-Block system additional feature functions as defined in §4.1 were computed on top of the letter-level alignment obtained from the baseline system. A log-linear model combining these features was learned with the gold-blocks described in §4.3. Transliteration blocks were extracted using the local-search §4.4. The other components remained the same as in the baseline system.

Table 2: Transliteration accuracy for different transliteration models.

System	Accuracy
Baseline	39.18%
L-Block	41.24%
LCBE	46.39%

The LCBE system is an extension to both the baseline and the L-Block system. The key difference in LCBE is that our proposed bi-stream HMM in Eqn. 2 was applied in both directions with extended letter-classes. The resulting combined alignment was used together with all features of the L-Block system to guide the local-search for extracting the blocks. The same procedure of decoding was then carried out for the unseen NEs using the extracted blocks.

To build the letter language model for the decoding process, we first split the English entities into characters; additional *position indicators* "_begin" and "_end" were added to the begin and end position of the named-entity; "_middle" was added between the first name and last name. A letter-trigram language model with SRI LM toolkit (Stolcke, 2002) was then built using the target side (English) of NE pairs tagged with the above position information.

Table 2 shows that the baseline system gives an accuracy of 39.18%, while the extended systems L-Block and LCBE give 41.24% and 46.39%, respectively. These results show that the additional features besides the letter-alignment are helpful. The L-Block system, which uses these features, outperforms the baseline system significantly by 2.1% absolute in accuracy. The results also show that the bi-stream HMM alignment, which uses not only the letters but also the letter-classes, leads to significant improvement. It outperforms the L-Block system, which does not leverage the letter-classes and monotone alignment, by 4.15% absolute.

5.3 Incorporation of Spell Checking

Our spelling-checker is based on the suggested word-forms from web search engines for ambiguous candidates. We collected web statistics frequency for both the proposed transliteration candidates from our system, and also the suggested candidates from web-search engines. All the candidates were re-ranked by their frequencies.

Figure 3 shows the performances on the held-out set, using system LCBE augmented with a spell-checker (*LCBE+Spell*), with varying sizes of N-best hypotheses lists. The held-out set contains 540 unique named entity pairs. We show accuracy when exact match is requested and when an edit distances of one is allowed.

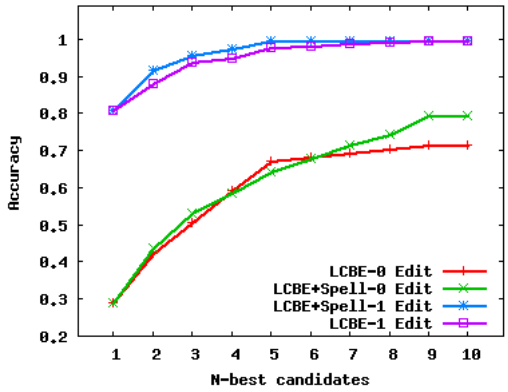

Figure 3: Transliteration accuracy of LCBE and LCBE+Spell models for 540 named entity pairs in the held-out set.

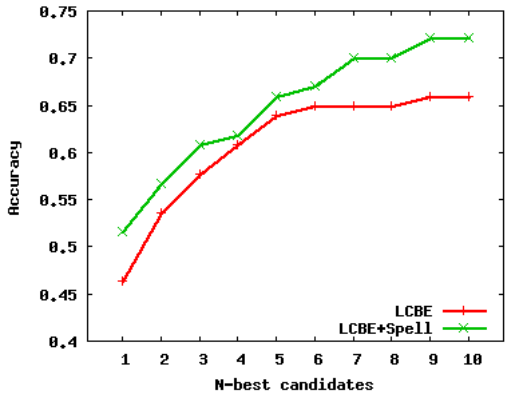

Figure 4: Transliteration accuracy of N-best hypotheses for LCBE and LCBE+Spell models it the MT-03 test set.

Figure 4 shows the performances in the unseen test set of LCBE and LCBE+Spell, with varying sizes of N-best hypotheses lists. LCBE+Spell reaches 52% accuracy in 1-best hypothesis. In the 5-best and 10-best cases, the accuracies of LCBE+Spell system archive the highest performances with 66% and 72.16% respectively. The spell-checker increases the 1-best accuracy by 11.12% and the 10-best accuracy by 7.69%. All these improvements are statistically significant. These results are also comparable to other state-of-the-art statistical Arabic name transliteration systems such as (Al-Onaizan and Knight, 2002).

5.4 Comparison with the Google Web Translation

We finally compared our best system with the state-of-the-art Arabic-English Google Web Translation (Google). Table 3 shows transliteration examples from our best system in comparison with Google (as in June 20, 2006)[3]. The Google system achieved 45.36% accuracy for the 1-best hypothesis, which is comparable to the results when using the LCBE transliteration system, while LCBE+Spell archived 52%.

[3]http://www.google.com/translate_t

Table 3: Transliteration examples between LCBE+Spell and Google web translation.

Source	Reference	LCBE+Spell	Google
سوساى	Sumaye	Sumaye	Somai
هازوميتسو	Hazumitsu	Hazumitsu	Hazoumitso
يلاه	Yalahow	Ylahu	Elaho
نكباخت	Nikbakht	Nkbakht	Nkbacht
ميكوياس	Mikulas	Mikulas	Mikoias
كومارانونج	Kumaratunga	Kumaratunga	Kumaratung
همدان	Hamdan	Hamdan	Hamedan
لُمازاندار ان	Mazandaran	Mazandaran	Mazandaran
ويكرمسينغه	Wickremasinghe	Wikramsinghe	The Ekermsingh

6 Conclusions and Discussions

In this paper we proposed a novel transliteration model. Viewing transliteration as a translation task we adopt alignment and decoding techniques used in a phrase-based statistical machine translation system to work on letter sequences instead of word sequences. To improve the performance we extended the HMM alignment model into a bi-stream HMM alignment by incorporating letter-classes into the alignment process. We also showed that a block-extraction approach, which uses a log-linear combination of multiple alignment features, can give significant improvements in transliteration accuracy. Finally, spell-checking based on work occurrence statistics obtained from the web gave an additional boost in transliteration accuracy.

The goal for this work is to improve the quality of machine translation, esp. when used in cross-lingual information retrieval and distillation tasks, by incorporating the proposed framework to handle unknown words. Figure 5 gives an example of the difference named entity transliteration can make. Shown are the original SMT system output, the translation when the proposed transliteration models are used to translate the unknown named-entities, and the reference translation. A comparison of the two SMT outputs indicates that integrating the proposed transliteration model into our machine translation system can significantly improve translation utility.

Acknowledgment

This work was partially supported by grants from DARPA (GALE project) and NFS (Str-Dust project).

References

Nasreen AbdulJaleel and Leah Larkey. 2003. Statistical transliteration for English-Arabic cross language information retrieval. In *Proceedings of the 12th International Conference on Information and Knowledge Management*, New Orleans, LA, USA, November.

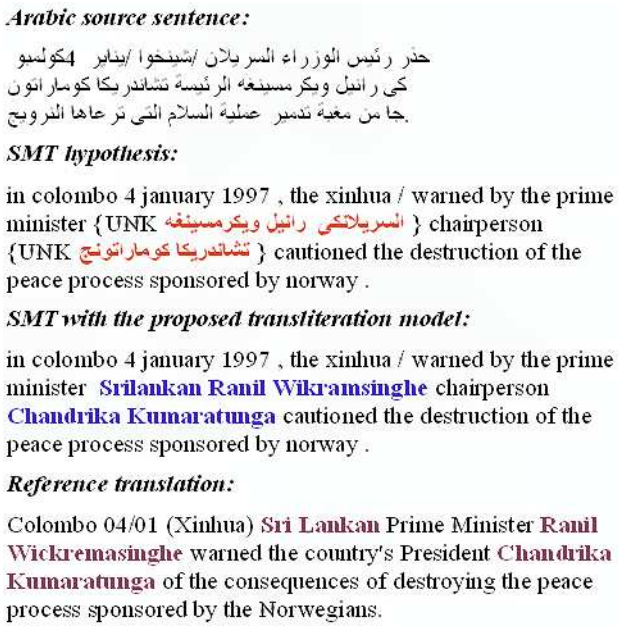

Figure 5: Incorporation of the transliteration model to our SMT System.

Yaser Al-Onaizan and Kevin Knight. 2002. Machine transliteration of names in Arabic text. In *Proceedings of ACL Workshop on Computational Approaches to Semitic Languages*, Philadelphia, PA, USA.

Mansur Arbabi, Scott M. Fischthal, Vincent C. Cheng, and Elizabeth Bart. 1994. Algorithms for Arabic name transliteration. In *IBM Journal of Research and Development*, volume 38(2), pages 183–193.

Adam L. Berger, Vincent Della Pietra, and Stephen A. Della Pietra. 1996. A maximum entropy approach to natural language processing. In *Computational Linguistics*, volume 22 of *1*, pages 39–71, March.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. In *Computational Linguistics*, volume 19(2), pages 263–331.

J.N. Darroch and D. Ratcliff. 1972. Generalized iterative scaling for log-linear models. In *Annals of Mathematical Statistics*, volume 43, pages 1470–1480.

Asif Ekbal, S. Naskar, and S. Bandyopadhyay. 2006. A modified joint source channel model for machine transliteration. In *Proceedings of COLING/ACL*, pages 191–198, Australia.

Heidi J. Fox. 2002. Phrasal cohesion and statistical machine translation. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*, pages 304–311, Philadelphia, PA, July 6-7.

Isao Goto, Naoto Kato, Noriyoshi Uratani, and Terumasa Ehara. 2003. Transliteration considering context information based on the maximum entropy method. In *Proceedings of MT-Summit IX*, New Orleans, Louisiana, USA.

Sanjika Hewavitharana, Bing Zhao, Almut Silja Hildebrand, Matthias Eck, Chiori Hori, Stephan Vogel, and Alex Waibel. 2005. The CMU statistical machine translation system for IWSLT2005. In *The 2005 International Workshop on Spoken Language Translation*.

Fei Huang. 2005. Cluster-specific name transliteration. In *Proceedings of the HLT-EMNLP 2005*, Vancouver, BC, Canada, October.

Kevin Knight and Jonathan Graehl. 1997. Machine transliteration. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL)*, Madrid, Spain.

Philipp Koehn. 2004. Pharaoh: a beam search decoder for phrase-based smt. In *Proceedings of the Conference of the Association for Machine Translation in the Americans (AMTA)*, Washington DC, USA.

Haizhou Li, Min Zhang, and Jian Su. 2004. A joint source-channel model for machine transliteration. In *Proceedings of 42nd ACL*, pages 159–166, Barcelona, Spain.

Franz J. Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. In *Computational Linguistics*, volume 1:29, pages 19–51.

Franz J. Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. In *Computational Linguistics*, volume 30, pages 417–449.

Jong-Hoon Oh and Key-Sun Choi. 2002. An English-Korean transliteration model using pronunciation and contextual rules. In *Proceedings of COLING-2002*, pages 1–7, Taipei, Taiwan.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proc. of the 40th Annual Conf. of the Association for Computational Linguistics (ACL 02)*, pages 311–318, Philadelphia, PA, July.

Bonnie Stalls and Kevin Knight. 1998. Translating names and technical terms in Arabic text. In *Proceedings of the COLING/ACL Workshop on Computational Approaches to Semitic Languages*, Montreal, Quebec, Canada.

Andreas Stolcke. 2002. SRILM – An extensible language modeling toolkit. In *Proc. Intl. Conf. on Spoken Language Processing*, volume 2, pages 901–904, Denver.

Paola Virga and Sanjeev Khudanpur. 2003. Transliteration of proper names in cross-lingual information retrieval. In *Proceedings of the ACL Workshop on Multi-lingual Named Entity Recognition*, Edmonton, Canada.

Stephan. Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM based word alignment in statistical machine translation. In *Proc. The 16th Int. Conf. on Computational Lingustics, (COLING-1996)*, pages 836–841, Copenhagen, Denmark.

Bing Zhao and Alex Waibel. 2005. Learning a log-linear model with bilingual phrase-pair features for statistical machine translation. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, Jeju Island, Korean, October.

Bing Zhao, Eric P. Xing, and Alex Waibel. 2005. Bilingual word spectral clustering for statistical machine translation. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, pages 25–32, Ann Arbor, Michigan, June.