English to Persian Transliteration

Sarvnaz Karimi, Andrew Turpin, and Falk Scholer

School of Computer Science and Information Technology
RMIT University, GPO Box 2476V, Melbourne 3001, Australia
{sarvnaz, aht, fscholer}@cs.rmit.edu.au

Abstract. Persian is an Indo-European language written using Arabic script, and is an official language of Iran, Afghanistan, and Tajikistan. Transliteration of Persian to English—that is, the character-by-character mapping of a Persian word that is not readily available in a bilingual dictionary—is an unstudied problem. In this paper we make three novel contributions. First, we present performance comparisons of existing grapheme-based transliteration methods on English to Persian. Second, we discuss the difficulties in establishing a corpus for studying transliteration. Finally, we introduce a new model of Persian that takes into account the habit of shortening, or even omitting, runs of English vowels. This trait makes transliteration of Persian particularly difficult for phonetic based methods. This new model outperforms the existing grapheme based methods on Persian, exhibiting a 24% relative increase in transliteration accuracy measured using the top-5 criteria.

1 Introduction

Translating words of a text from a *source* language into a different *target* language can be efficiently achieved using a bilingual vocabulary, where every source word has a counterpart in the target language. In practice, however, there are often out-of-vocabulary (OOV) words that do not appear in the source dictionary. This is particularly common for proper nouns such as company, people, place and product names. In these cases *transliteration* must occur, where the OOV word is spelled out in the target language using character based methods.

Automatic transliteration of English OOV words has been studied for several languages, including Arabic [1], Korean [7,12], Chinese [14], Japanese [2,8,12], and the romantic languages [10,13]. Transliteration between English and Persian has not been previously studied.

Persian, also known as Farsi, is one of the oldest Indo-European languages, and is the official language of Iran, Afghanistan, and Tajikistan. Unlike many of the Indo-European languages (for example, English) it is not written in the Latin/Roman alphabet, but uses the same script as Arabic, with four additional characters: پ, چ, ژ, گ. So while at first glance it may seem that transliteration techniques that work for Arabic should work for Persian, the roots of the two languages are very different, and the phonetic interpretation of the Arabic script differs between the two languages. For example, ت and ط are pronounced

F. Crestani, P. Ferragina, and M. Sanderson (Eds.): SPIRE 2006, LNCS 4209, pp. 255–266, 2006.

differently in Arabic, but both are pronounced as "t" (as in "Peter") in Persian. If an Arabic name is being written in Persian, it is more likely that ط be preferred to ت; for words from other languages, authors may have an independent preference for either ت or ط, as they convey the same phonetic sound. These *redundant* characters make the transliteration of Persian a new and interesting problem to study.

Previous transliteration methods can be grouped into three main categories: grapheme-based approaches; phonetic-based approaches; and approaches that combine features of both. Grapheme-based (or *direct*) methods map groups of characters in the source word S directly to groups of characters in the target word T. Phonetic (or *pivot*) methods, on the other hand, first try and identify phonemes in the source word S, and then map those phonemes to character representations in the target language to get T, the target word. As the phonetic approaches have extra steps, they are in general more error prone than their grapheme-based counterparts, and typically success rates of phonetic based only approaches are lower than combined methods [2,12]. Given that grapheme based methods tend to be more successful than phoneme based methods, and that Persian words often omit vowels that would appear in their English counterparts making phonetic matching difficult, we adopt a grapheme based approach.

In this paper, we present results for transliterating English to Persian using a basic grapheme model as has been proposed for Arabic [1]. We extend this approach using more context characters from the source word S, and then introduce a more complex model to deal with some specifics of the Persian language. On a corpus of 1,857 proper names, our improved system achieves 66% accuracy when only one candidate transliteration is generated, compared with the baseline system at 48%. This is the first successful English to Persian transliteration system reported in the literature. We also discuss the problems in constructing an OOV corpus for Persian-English, and hypothesise that these problems will re-occur in many transliteration studies.

2 Background

In general, grapheme based transliteration is a three stage process.

Alphabet selection. In the first stage, an alphabet of source and target symbols, Σ_S and Σ_T, is created from a training corpus of known source-target word pairs. Note that symbols may be individual characters, or groups of characters, in the original alphabets. For example, the pair of English characters "ch" might be included as a single symbol in the source alphabet Σ_S. Symbol alphabets can be derived using statistical methods such as hidden Markov models [7] or translation models [3]. These have been implemented in the GIZA++ tool [11], which we use to obtain symbol alphabets for our experiments. An alternative approach

is to hand-craft the alphabets of allowable source and target graphemes, for example as adopted by AbdulJaleel and Larkey [1].

Probability generation. In the second stage, the training corpus of known source-target word pairs is used to collect statistics on the frequency of occurrence of chosen alphabet symbols. In general, we wish to compute the probability $P(t_i|s_i, c_i)$, which represents the probability of generating target symbol $t_i \in \Sigma_T$ given the source symbol $s_i \in \Sigma_S$ in the context c_i, which is a string drawn from $(\Sigma_T \cup \Sigma_S)$. In the most basic model, no context is used, so $c_i = \epsilon$, the empty string, and the transliteration is given simply by $P(t_i|s_i)$.

Use of context has been shown to improve transliteration accuracy. In his study of transliterating six romantic languages, Linden [10] use the two previous source symbols and one following source symbol as context to get $P(t_i|s_i, c_i = s_{i-2}s_{i-1}s_{i+1})$, which gave an improvement of 69% over a baseline technique. In similar work on Korean, Jung et al. [7] proposed taking advantage of past and future source symbols, and one past target language symbol, to compute $P(t_i|s_i, c_i = s_{i-2}s_{i-1}s_{i+1}t_{i-1})$, which gave a 5% improvement over their baseline.

In all methods that employ a non-empty context c_i, provision must be made for *backoff* to a smaller context. For example, if attempting to transliterate the symbol "o" in the word "lighthouse" in the context $c_i = s_{i-4}s_{i-3}s_{i-2}s_{i-1} =$ "ghth", it is likely that the context "ghth" occurred very infrequently in the training corpus, and so statistics derived in this context may not be reliable. In this case, a backoff scheme may try the context $c_i = s_{i-3}s_{i-2}s_{i-1} =$"hth", which again may not occur frequently enough to be trusted, and so the next backoff context must be tried. This backoff method is used in the PPM data compression scheme [4].

Transliteration. The third stage of transliteration parses the source word S into symbols from Σ_S and computes, for possible target words T,

$$P(T|S) = P(T) \prod_{i=1}^{|T|} P(t_i|s_i, c_i), \tag{1}$$

where the first term $P(T)$ is a *target language model*, that is, the probability of the target word T appearing independently of any source information. For example, the word "zxqj" is extremely unlikely to appear as the English transliteration of any word in any language, and so $P($"zxqj"$)$ would be very small. Probability $P($"the"$)$, on the other hand, may sit at around 0.01. Distribution $P(T)$ can be computed using a *zero-order* model from the training corpus as $P(T) = \prod_{i=1}^{|T|}(\text{freq}(t_i)/D)$, where D is the total number of characters in the training corpus; and freq gives the frequency of its argument in the training corpus. In our experiments below we used a *first-order* model where $P(T) = \prod_{i=2}^{|T|}(\text{freq}(t_{i-1}t_i)/\text{freq}(t_i))$.

The target words can then be sorted by $P(T|S)$, given in Equation 1, to give a ranked list of the most likely transliterations of S.

3 Transliteration Methods

In this section we describe our implementation of the transliteration process. First we introduce baseline systems, followed by an explanation of our novel transliteration approach, where contexts respect vowel-consonant boundaries.

3.1 Baseline Systems

Alphabet selection. In order to select the source and target alphabets, Σ_S and Σ_T, we made use of the character-level alignment of source and target word pairs produced by GIZA++ [11]. If this resulted in a group of characters mapping to a single character in either direction, then the group was added as a single symbol to Σ_S or Σ_T, as required. Consider an example, where the word "stella" is aligned between Persian and English characters:

Source (English):	s	t	e	ll	a
Target (Persian, left-to-right):	سا	ت	ϵ	ل	ا

Here, both "te" and "ll" would be added to Σ_S, as they map to a single Persian character (ϵ represents a null character). The symbol "سا" would be added to Σ_T as these two Persian characters align with only one English character.

Probability generation. Previous Arabic transliteration systems use an empty context [1], whereas successful transliteration systems for romantic languages (arguably more close to Persian, in spite of script) use $c_i = s_{i-2}s_{i-1}s_{i+1}$ for transliterating s_i [10]. Accordingly, we tested a variety of contexts based on past and future source symbols.

For brevity, we introduce the notation $n\backslash m$ to indicate that n previous source symbols and m future source symbols make up a context, that is:

$$c_i = n\backslash m = s_{i-n} \dots s_{i-1}s_{i+1} \dots s_{i+m}$$

In order to avoid boundary conditions, we also assume that S is extended to the left and right as far as is required with a special symbol that always transliterates to an empty symbol ϵ. Therefore, $[-n, \dots, |S| + m]$ are all valid indexes into S.

During this probability generation phase, frequency counts for each mapping contained in the aligned words of the training set are gathered. We define

$$f(t_i, s_i, n, m) = \text{frequency of } t_i \text{ aligning with } s_i \text{ in context } n\backslash m.$$

Once all frequencies are gathered, they are simply normalised into probabilities to get

$$P(t_i|s_i, c_i = n\backslash m) = f(t_i, s_i, n, m) \Big/ \sum_{\forall j} f(t_j, s_i, n, m).$$

1) Let ℓ be the length of the longest context in characters to the left,
   r be the length of the longest context in characters to the right, and
   $S = [s_1 \dots s_{|S|}]$ be the input word made up of $|S|$ characters.
2) Set $X \leftarrow$ the empty set.
3) For $i \leftarrow 1$ to $|S|$ do
4) Let $p \leftarrow \ell$ and $q \leftarrow r$.
5) While $p > 0$, $q > 0$ and context $S[i - p \dots i + q]$ has a frequency $< M$
6) Set $p \leftarrow p - 1$ and set $q \leftarrow q - 1$.
7) If $p > 0$ or $q > 0$ then
8) Append all possible transliterations of s_i in context
    $S[i - \max(p, 0) \dots i + \max(q, 0)]$ to the elements of X
    and record the associated probabilities of resulting words.
9) else
    Append all possible transliterations of s_i without context
    to the elements of X and record associated probabilities.
10) Sort X and output transliterations.

Fig. 1. The transliteration algorithm.

Transliteration. While source symbols are readily identified in the aligned words of the training corpus, when presented with a source word alone for transliteration there may be several parsings possible using elements from the source alphabet Σ_S. One possible approach is to greedily choose the longest matching source symbol from left-to-right, and then use this parsing throughout the transliteration process [1]. An alternate is to define a *source language model* which will apply probabilities to various parsings, and work this probability into Equation 1 for $P(T|S)$ [7].

The approach we have taken here is to not commit to a single parsing; instead, we process the source character by character. Similarly the backoff process for contexts is on a character by character basis, rather than a symbol by symbol basis. Note that this requires us to keep track of the number of characters generated in the longest context of the probability generation phase; while it is known that this context will contain $n + m$ symbols, this will map to $\ell + r$ characters, where ℓ is the maximum number of characters in the n previous symbols (to the ℓeft), and r is the maximum number of characters in the m future symbols (to the $r$ight).

A high level description of the transliteration algorithm is given in Figure 1. An attempt is made to transliterate each character in the longest possible context, with the context reducing by one character each time at both ends until the context occurs at least M times in the training data (Steps 5 and 6). Once a context is found, all possible target symbols t_i that arise from the character s_i in that context are incorporated into the set of transliterations so far, X. If a context is not found, then a straight mapping of the individual character without context is carried out (Step 9). Since there are $|\Sigma_T|^{|T|}$ possible target words, it is not practical to exhaustively generate all variants as is shown in Steps 8 and 9. In our implementation we use the k-shortest paths algorithm due to Dijkstra [5],

which greedily expands prefixes of strings in X with the highest probabilities, and therefore guarantees that the k highest scoring transliterations are found.

There are several alternate implementations of Step 5 and 6 for reducing contexts. The approach we describe in Figure 1, where both past (p) and future (q) horizons are shrunk, we call BACKOFF-A. The second approach with which we experimented was first fully reducing future contexts, and only when q reaches zero do we begin to reduce past contexts (p). We label this approach BACKOFF-B. In all experiments we used $M = 2$.

3.2 Collapsed-Vowel Models

Purely selecting contexts according to a $n\backslash m$ model as described in the previous section ignores any characteristics that we may know of the source and target languages. For instance, Persian speakers tend to transliterate diphthongs (vowel sounds that start near the articulatory position for one vowel and moves toward the position for another) of other languages to monophthongs (two written vowels that represent a single sound). In addition, short vowel sounds in English are often omitted altogether in Persian. This suggests a model where runs of English vowels are transliterated in the context of their surrounding consonants. Rather than employ a full natural language analysis technique, as has been done for Chinese [14], we opt for a simpler segmentation approach and propose the following *collapsed-vowel* models.

Alphabet selection. We define Σ_S and Σ_T as in the baseline systems: composite symbols are identified by aligning words using the statistical translation model implemented in GIZA++.

Once this initial parsing is complete, we re-segment the words using consonant and vowel boundaries to define new symbols that we add to the existing alphabets. Each source word is parsed into a sequence of single consonants, represented by C, and runs of consecutive vowels represented by V. These are then grouped into *segments* in one of four ways:

1. VC, a run of vowels at the beginning of a word followed by a consonant;
2. CC, two consonants not separated by any vowels;
3. CVC, a run of vowels bounded by two consonants; and
4. CV, a run of vowels at the end of a word preceded by a consonant.

The first consonant in each segment overlaps the final consonant in the preceding segment. In each case the full composite symbol is added to the source alphabet (to later be used as context), and the composite symbol without the tailing consonant is added to both alphabets (to be used for transliteration).

For example, consider the word "baird". This would first be parsed as:

Source (English):	b	ai	r	d
Target (Persian, left-to-right):	ب	ی	ر	د

resulting in "ai" being added to Σ_S (as in the baseline system).

i	S	Segment	Method CV $s_i \vert c_i$	Method CV-BROAD $s_i \vert c_i$
1	e n	VC	e|en	e|eC
2	n r	CC	n|nr	n|CC
3	r i q	CVC	ri|riq	r|C and i|CiC
4	q u e	CV	que|que	q|C and ue|Cue

Fig. 2. Example of the segments, symbols and contexts generated for the source word "enrique"

In the second phase, the word would be parsed into segments as:

Source (English):	b	ai	r	d
Target (Persian, left-to-right):	ب	ی	ر	د
Segmentation:	C	V	C	C
	Segment 1			
			Segment 2	

resulting in the possible contexts "bair" and "rd" being added into Σ_S as a CVC and CC segments respectively. The additional source symbol "bai", derived by dropping the C from CVC is added to Σ_S, and "r" is already in Σ_S. In the target alphabet, ی ب is added as a truncated CVC target symbol, and the single character "ر" is already in Σ_T.

Probability generation. Probabilities are generated from frequency counts in the training data as for the baseline systems, but the contexts are now chosen according to segments, rather than runs of symbols (graphemes). We use two techniques for selecting contexts. In the first, labeled CV, each segment forms a context for the symbol that prefixes the segment. For example, the fourth column of Figure 2 shows how the contexts and symbols for the source word "enrique" are derived.

In the second technique for choosing contexts, CV-BROAD, we relax the strict matching on the consonant component of the context so that it can match all consonants, thus increasing the amount of training data for a context. In turn, this leads to rare, but existent, transliterations that would otherwise not be available. Vowels are only used in a context when transliterating a symbol that contains a vowel. This is shown in the final column of Figure 2 for the example string "enrique".

The backoff approach for each of the four segments is given by the following three rules, which are applied in order:

1. if the context is a CV segment, and does not occur at least M times in the training data, separate the symbol into a C then V context; else
2. if the context is a V segment representing a run of r vowels, and does not occur at least M times in the training data, reduce the length of the run to $r-1$ and transliterate the final vowel as a single character with no context; else

3. for all other contexts, if it occurs less than M times in the training data, drop the tailing C and try this shorter context.

Transliteration. The transliteration proceeds as described for the baseline systems in Section 3.1, where Dijkstra's k-shortest path is used to generate the k highest scoring transliterations.

4 Experiments

In this section we describe our experimental framework, including the data used for training and testing purposes, the transliteration models that are evaluated, and metrics used to quantify performance.

4.1 Data

The first two stages of transliteration require a training corpus of source and target word pairs. An English-Persian corpus was developed by taking 40,000 phrases from lists of English names that occurred on the World Wide Web. These names were names of geographical places, persons and companies, extracted from different publicly available resources. This English corpus was then transliterated by 26 native Persian speakers. Redundant and allophone characters were often used by the transliterators for Arabic words of the collection. Depending on the transliterator's assumptions about the origin of a word, different characters may have been chosen for the same word. For example, if the transliterator assumed "John Pierre" was French, then for the character "j" the transliterator might have employed the rare Persian character ژ ,pronounced same as "su" in measure, instead of widely used ج which sounds as "j" in jet. Finally the corpus was split into words, and the unique word pairs extracted, resulting in 16,952 word pairs. We refer to this collection as ENGLISH+.

The corpus selection process did not guarantee that all included words were of English origin. As a result, there are many words in the corpus that are already transliterated from other languages such as French, Arabic, and Chinese. We randomly chose a subset of 2,000 name pairs from the ENGLISH+ collection, from which all names with origins from non-English script languages were removed. This resulted a sub-collection of 1,857 name pairs, labelled ENGLISH.

In our experiments we apply ten-fold cross-validation; so for any one run, 90% of a collection is used as training data, with the remaining 10% being used as test data, and results are averages over the 10 possible runs.

4.2 Models

In our experiments we consider baseline transliteration models using a variety of context sizes and backoff options, as well as our novel collapsed-vowel models.

For the baseline models, we tested combinations of past (n) and future (m) contexts $n\backslash m$, using the following settings: 0\0, 1\0, 2\0, 0\1, 0\2, and 2\1. These contexts were all run with backoff method BACKOFF-B and a uniformly distributed target language model. Also included is the first-order target language model described in Section 2 for 0\0 and 1\0, denoted as 0\0T and 1\0T. We also experimented with BACKOFF-A for contexts 1\1, 1\1 and 2\1 using a uniform target language model, denoted 1\1A, 2\2A and 2\1A.

These models were run on both the ENGLISH and ENGLISH+ collections.

4.3 Metrics

The results of our transliteration experiments are evaluated using the standard measure of word accuracy, based on the edit distance between the transliterated word and the correct word. The edit distance measures the number of character insertions, deletions and substitutions that are required to transform one word into another [9].

Word accuracy (WA), also known as transliteration accuracy, measures the proportion of transliterations that are correct:

$$WA = \frac{Number\ of\ transliterations\ with\ ED{=}0}{Total\ number\ of\ test\ words}$$

where, ED is the edit distance between two strings [6]. WA is reported for different cutoff values. For example, *top-1* WA indicates the proportion of words in the test set for which the correct transliteration was the first candidate answer returned by the transliteration system, while *top-5* indicates the proportion of words for which the correct transliteration was returned within the first five candidate answers.

Where statistical tests are performed on accuracy, the test employed is a paired t-test on the accuracy-percentages calculated for each step of the 10-fold cross validation runs.

5 Results

In this section we present the results of our experiments into the effects of context size, target language models, and collapsed vowel models. In the tables that follow, the results of the most accurate methods are highlighted in bold.

Past and Future Context. The results of using varying past (n) and future (m) context windows are shown in Table 1. The highest performance is achieved using a context window of 1\0, that is, using a history of just one character. The difference in performance between this context and using no context is statistically significant ($p < 0.002$ for *top-1*).

Extending the historical context causes performance to deteriorate, compared to using just one historical symbol. Similarly, the addition of future context causes performance to fall when past context is also present. Using the symmetrical backoff method, BACKOFF-A, rather than BACKOFF-B, improves performance

Table 1. Mean word accuracy (%) for changing past and future context sizes

		0\0	1\0	2\0	0\1	0\2	1\1*A*	2\2*A*	2\1	2\1*A*
ENGLISH	*Top-1*	47.8	**59.2**	49.5	43.1	54.6	54.5	27.6	47.8	38.4
	Top-5	73.7	**84.9**	73.9	70.5	76.3	79.6	41.7	72.3	65.0
	Top-10	79.9	**89.3**	77.4	75.1	78.8	83.3	46.1	78.0	71.9
ENGLISH+	*Top-1*	9.0	**45.7**	15.1	32.6	12.8	31.7	6.5	9.3	22.5
	Top-5	11.7	**71.3**	23.6	60.1	21.6	55.7	14.1	14.9	41.9
	Top-10	16.6	**77.2**	26.2	68.5	23.6	63.2	17.3	16.9	51.1

Table 2. Mean word accuracy (%) when using a target language model (T) and collapsed vowel schemes (CV and CV-BROAD)

		0\0	0\0*T*	1\0	1\0*T*	CV	CV+
ENGLISH	*Top-1*	47.8	34.8	59.2	55.1	**66.4**	64.9
	Top-5	73.7	45.1	84.9	79.7	84.4	**87.3**
	Top-10	79.9	50.6	89.3	84.3	88.5	**95.0**
ENGLISH+	*Top-1*	9.0	3.3	45.7	40.8	**50.1**	49.5
	Top-5	11.7	4.6	71.3	63.6	74.0	**80.8**
	Top-10	16.6	5.2	77.2	69.3	79.4	**88.0**

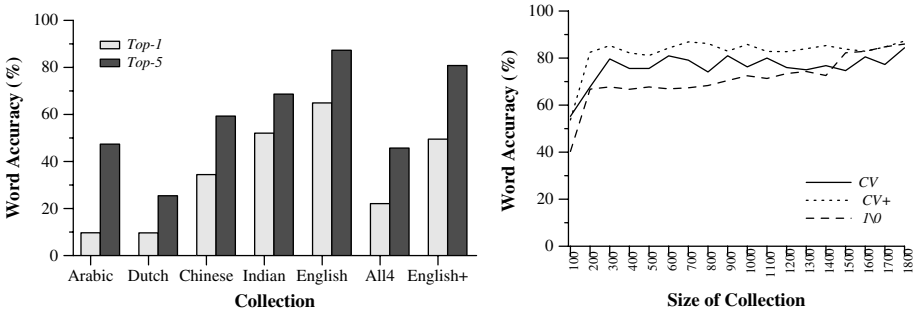
for the larger ENGLISH+ collection when both past and future context is used in the transliteration model. However, overall performance here is still significantly worse than when using a context of 1\0 ($p < 0.005$).

Target Language Models. The effect of introducing a target language model is shown in Table 2; schemes using a first-order target language model are labeled with a "T". It can be seen that using a target language model has a detrimental effect on transliteration performance for both 0\0 and 1\0 contexts.

Collapsed Vowel Approaches. The results for our novel collapsed vowel models are shown in the final columns of Table 2, labeled CV and CV-BROAD. The CV scheme shows the best performance for *top-1* transliteration for both the ENGLISH and ENGLISH+ collections, an improvement that is statistically significant over the best 1\0 baseline ($p < 0.008$). For *top-5* and *top-10* word accuracy, the CV-BROAD model shows highest performance.

Language-based Effects. The large differences between the results on the ENGLISH and ENGLISH+ collections suggest that, although a larger training collection could assist the transliteration process, the origin of the source language could have a deleterious impact on accuracy.

Figure 5(a) shows the accuracy of the CV approach on different sub-collections that we extracted from the ENGLISH+ collection. A word was assigned to a country according to its appearance on WWW pages of that country. For example, "Groenendijk" appears in many pages of the domain `.nl`, and so was assigned



(a) Different language origins (Top-1) (b) Different sizes (Top-5)

Fig. 3. Mean word accuracy of different collections

to the Dutch category. Note that the accuracy is substantially reduced for all languages individually and altogether ("All4").

Collection Size Effects. One possible reason for CV performing poorly on the country based data sets is that they are small datasets. To investigate this possibility, we randomly partitioned ENGLISH into sets of increasing size and computed the accuracy of CV on those partitions.

The results are shown in Figure 5(b). As can be seen, any data set above 200 words performed well, and so it would seem that the small size of the individual country datasets (all are larger than 200) is not to blame. There appears to be a genuine difference between the performance of CV on native English and non-English words that have already been transliterated from another language.

6 Conclusions and Future work

In this paper, we have investigated a variety of transliteration techniques and their effectiveness in transliterating words from English to Persian. We investigated the performance of existing grapheme-based approaches, and examined a range of context and target language model options. The use of context is important: using a small historical context (one past symbol) makes transliteration significantly more effective than using no context. However, using larger context sizes, or using future context, has a detrimental effect. This effect is surprising; in general a larger context should improve the predictive power of the model. The effect may be due in part to our symbol parsing approach. In future work, we plan to investigate whether incorporating a full source language model into our transliteration process can extend the range of useful contexts.

A new transliteration model of Persian, based on collapsing runs of vowels, was introduced and evaluated. Our novel techniques increased accuracy over standard grapheme-based approaches with no context by around 18% on a carefully chosen native English data set (66% up from 48% for *top-1*), and by 41% on a more general English data set that was not filtered (50% up from 9% for *top-1*). In

future work, we will evaluate our new models on other languages with similar vowel transliteration patters as Persian, such as Arabic and Indonesian.

Investigating the effects of including different source languages in training data demonstrated that this introduces noise, and can have a substantial negative impact on transliteration performance. Previous work has generally avoided this problem by using small test sets that are carefully controlled. As far as we are aware, this is the first work on transliteration that has examined the effects of data quality on accuracy. Our future research will be to further investigate and quantify the noise that is generated as part of this phenomenon.

Acknowledgments

This work was supported in part by Australian Research Council Grant DP055-8916 (AT) and the Australian government IPRS program (SK).

References

1. Nasreen AbdulJaleel and Leah S. Larkey. Statistical transliteration for English-Arabic cross language information retrieval. In *CIKM*, pages 139–146, 2003.
2. Slaven Bilac and Hozumi Tanaka. Direct combination of spelling and pronunciation information for robust back-transliteration. In *CICLing*, pages 413–424, 2005.
3. Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computional Linguistics*, 19(2):263–311, 1993.
4. John G. Cleary and Ian H. Witten. A comparison of enumerative and adaptive codes. *IEEE Transactions on Information Theory*, 30(2):306–315, 1984.
5. David Eppstein. Finding the k shortest paths. *SIAM J. Computing*, 28(2):652–673, 1998.
6. Patrick A. V. Hall and Geoff R. Dowling. Approximate string matching. *ACM Comput. Surv.*, 12(4):381–402, 1980.
7. Sung Young Jung, Sung Lim Hong, and Eunok Paek. An English to Korean transliteration model of extended markov window. In *COLING*, pages 383–389, 2000.
8. Kevin Knight and Jonathan Graehl. Machine transliteration. *Computational Linguistics*, 24(4):599–612, 1998.
9. Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Doklady Akademii Nauk SSSR*, 163(4):845–848, 1965.
10. Krister Linden. Multilingual modeling of cross-lingual spelling variants. *Inf. Retrieval*, 9(3):295–310, 2005.
11. Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, 2003.
12. Jong-Hoon Oh and Key-Sun Choi. An ensemble of transliteration models for information retrieval. *Inf. Process. Manage.*, 42(4):980–1002, 2006.
13. Jarmo Toivonen, Ari Pirkola, Heikki Keskustalo, Kari Visala, and Kalervo Järvelin. Translating cross-lingual spelling variants using transformation rules. *Inf. Process. Manage.*, 41(4):859–872, 2005.
14. Stephen Wan and Cornelia Verspoor. Automatic English-Chinese name transliteration for development of multilingual resources. In *COLING-ACL*, pages 1352–1356, 1998.