Collapsed Consonant and Vowel Models: New Approaches for English-Persian Transliteration and Back-Transliteration

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

Abstract

We propose a novel algorithm for English to Persian transliteration. Previous methods proposed for this language pair apply a word alignment tool for training. By contrast, we introduce an alignment algorithm particularly designed for transliteration. Our new model improves the English to Persian transliteration accuracy by 14% over an n-gram baseline. We also propose a novel back-transliteration method for this language pair, a previously unstudied problem. Experimental results demonstrate that our algorithm leads to an absolute improvement of 25% over standard transliteration approaches.

1 Introduction

Translation of a text from a source language to a target language requires dealing with technical terms and proper names. These occur in almost any text, but rarely appear in bilingual dictionaries. The solution is the transliteration of such *out-of-dictionary* terms: a word from the source language is transformed to a word in the target language, preserving its pronunciation. Recovering the original word from the transliterated target is called back-transliteration. Automatic transliteration is important for many different applications, including machine translation, cross-lingual information retrieval and cross-lingual question answering.

Transliteration methods can be categorized into grapheme-based (AbdulJaleel and Larkey, 2003; Li et al., 2004), phoneme-based (Knight and Graehl, 1998; Jung et al., 2000), and combined (Bilac and Tanaka, 2005) approaches. Grapheme-based methods perform a direct orthographical mapping between source and target words, while phoneme-based approaches use an intermediate phonetic representation. Both grapheme- or phoneme-based methods usually begin by breaking the source word into segments, and then use a source segment to target segment mapping to generate the target word. The rules of this mapping are obtained by aligning already available transliterated word pairs (training data); alternatively, such rules can be handcrafted. From this perspective, past work is roughly divided into those methods which apply a word alignment tool such as GIZA++ (Och and Ney, 2003), and approaches that combine the alignment step into their main transliteration process.

Transliteration is language dependent, and methods that are effective for one language pair may not work as well for another. In this paper, we investigate the English-Persian transliteration problem. Persian (Farsi) is an Indo-European language, written in Arabic script from right to left, but with an extended alphabet and different pronunciation from Arabic. Our previous approach to English-Persian transliteration introduced the grapheme-based *collapsed-vowel* method, employing GIZA++ for source to target alignment (Karimi et al., 2006). We propose a new transliteration approach that extends the *collapsed-vowel* method. To meet Persian language transliteration requirements, we also propose a novel alignment algorithm in our training stage, which makes use of statistical information of

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 648–655,
Prague, Czech Republic, June 2007.

the corpus, transliteration specifications, and simple language properties. This approach handles possible consequences of elision (omission of sounds to make the word easier to read) and epenthesis (adding extra sounds to a word to make it fluent) in written target words that happen due to the change of language. Our method shows an absolute accuracy improvement of 14.2% over an n-gram baseline.

In addition, we investigate the problem of back-transliteration from Persian to English. To our knowledge, this is the first report of such a study. There are two challenges in Persian to English transliteration that makes it particularly difficult. First, written Persian omits short vowels, while only long vowels appear in texts. Second, monophthongization (changing diphthongs to monophthongs) is popular among Persian speakers when adapting foreign words into their language. To take these into account, we propose a novel method to form transformation rules by changing the normal segmentation algorithm. We find that this method significantly improves the Persian to English transliteration effectiveness, demonstrating an absolute performance gain of 25.1% over standard transliteration approaches.

2 Background

In general, transliteration consists of a training stage (running on a bilingual training corpus), and a generation – also called testing – stage.

The training step of a transliteration develops transformation rules mapping characters in the source to characters in the target language using knowledge of corresponding characters in transliterated pairs provided by an alignment. For example, for the source-target word pair (pat,پات), an alignment may map "p" to "پ" and "a" to "ا", and the training stage may develop the rule $pa \rightarrow$ ا, with "ا" as the transliteration of "a" in the context of "pa". The generation stage applies these rules on a segmented source word, transforming it to a word in the target language.

Previous work on transliteration either employs a word alignment tool (usually GIZA++), or develops specific alignment strategies. Transliteration methods that use GIZA++ as their word pair aligner (AbdulJaleel and Larkey, 2003; Virga and Khudanpur, 2003; Karimi et al., 2006) have based their work on the assumption that the provided alignments are reliable. Gao et al. (2004) argue that precise alignment can improve transliteration effectiveness, experimenting on English-Chinese data and comparing IBM models (Brown et al., 1993) with phoneme-based alignments using direct probabilities.

Other transliteration systems focus on alignment for transliteration, for example the joint source-channel model suggested by Li et al. (2004). Their method outperforms the noisy channel model in direct orthographical mapping for English-Chinese transliteration. Li et al. also find that grapheme-based methods that use the joint source-channel model are more effective than phoneme-based methods due to removing the intermediate phonetic transformation step. Alignment has also been investigated for transliteration by adopting Covington's algorithm on cognate identification (Covington, 1996); this is a character alignment algorithm based on matching or skipping of characters, with a manually assigned cost of association. Covington considers consonant to consonant and vowel to vowel correspondence more valid than consonant to vowel. Kang and Choi (2000) revise this method for transliteration where a skip is defined as inserting a null in the target string when two characters do not match based on their phonetic similarities or their consonant and vowel nature. Oh and Choi (2002) revise this method by introducing *binding*, in which many to many correspondences are allowed. However, all of these approaches rely on the manually assigned penalties that need to be defined for each possible matching.

In addition, some recent studies investigate discriminative transliteration methods (Klementiev and Roth, 2006; Zelenko and Aone, 2006) in which each segment of the source can be aligned to each segment of the target, where some restrictive conditions based on the distance of the segments and phonetic similarities are applied.

3 The Proposed Alignment Approach

We propose an alignment method based on segment occurrence frequencies, thereby avoiding predefined matching patterns and penalty assignments. We also apply the observed tendency of aligning consonants to consonants, and vowels to vowels, as a substitute for phonetic similarities. Many to many, one to many, one to null and many to one alignments can be generated.

3.1 Formulation

Our alignment approach consists of two steps: the first is based on the consonant and vowel nature of the word's letters, while the second uses a frequency-based sequential search.

Definition 1 *A bilingual corpus* $\mathcal{B}$ *is the set* $\{(S, T)\}$*, where* $S = s_1..s_\ell$*,* $T = t_1..t_m$*,* s_i *is a letter in the source language alphabet, and* t_j *is a letter in the target language alphabet.*

Definition 2 *Given some word,* w*, the consonant-vowel sequence* $p = (C|V)^+$ *for* w *is obtained by replacing each consonant with* C *and each vowel with* V*.*

Definition 3 *Given some consonant-vowel sequence,* p*, a reduced consonant-vowel sequence* q *replaces all runs of* C*'s with* $\mathcal{C}$*, and all runs of* V*'s with* $\mathcal{V}$*; hence* $q = q'|q''$*,* $q' = \mathcal{V}(\mathcal{C}\mathcal{V})^*(\mathcal{C}|\epsilon)$ *and* $q'' = \mathcal{C}(\mathcal{V}\mathcal{C})^*(\mathcal{V}|\epsilon)$*.*

For each natural language word, we can determine the *consonant-vowel* sequence (p) from which the *reduced consonant-vowel* sequence (q) can be derived, giving a common notation between two different languages, no matter which script either of them use. To simplify, semi-vowels and approximants (sounds intermediate between consonants and vowels, such as "w" and "y" in English) are treated according to their target language counterparts.

In general, for all the word pairs (S, T) in a corpus $\mathcal{B}$, an alignment can be achieved using the function

$$f : \mathcal{B} \rightarrow \mathcal{A}; (S, T) \mapsto (\hat{S}, \hat{T}, r).$$

The function f maps the word pair $(S, T) \in \mathcal{B}$ to the triple $(\hat{S}, \hat{T}, r) \in \mathcal{A}$ where $\hat{S}$ and $\hat{T}$ are substrings of S and T respectively. The frequency of this correspondence is denoted by r. $\mathcal{A}$ represents a set of substring alignments, and we use a per word alignment notation of a_{e2p} when aligning English to Persian and a_{p2e} for Persian to English.

3.2 Algorithm Details

Our algorithm consists of two steps.

Step 1 **(Consonant-Vowel based)**

For any word pair $(S, T) \in \mathcal{B}$, the corresponding *reduced consonant-vowel* sequences, q_S and q_T, are generated. If the sequences match, then the aligned consonant clusters and vowel sequences are added to the alignment set $\mathcal{A}$. If q_S does not match with q_T, the word pair remains unaligned in *Step 1*.

The assumption in this step is that transliteration of each vowel sequence of the source is a vowel sequence in the target language, and similarly for consonants. However, consonants do not always map to consonants, or vowels to vowels (for example, the English letter "s" may be written as "اس" in Persian which consists of one vowel and one consonant). Alternatively, they might be omitted altogether, which can be specified as the null string, ε. We therefore require a second step.

Step 2 **(Frequency based)**

For most natural languages, the maximum length of corresponding phonemes of each grapheme is a digraph (two letters) or at most a trigraph. Hence, alignment can be defined as a search problem that seeks for units with a maximum length of two or three in both strings that need to be aligned. In our approach, we search based on statistical occurrence data available from *Step 1*.

In *Step 2*, only those words that remain unaligned at the end of *Step 1* need to be considered. For each pair of words (S, T), matching proceeds from left to right, examining one of the three possible options of transliteration: single letter to single letter, digraph to single letter and single letter to digraph. Trigraphs are unnecessary in alignment as they can be effectively captured during transliteration generation, as we explain below.

We define four different valid alignments for the source ($S = s_1 s_2 \ldots s_i \ldots s_l$) and target ($T = t_1 t_2 \ldots t_j \ldots t_m$) strings: (s_i, t_j, r), $(s_i s_{i+1}, t_j, r)$, $(s_i, t_j t_{j+1}, r)$ and (s_i, ε, r). These four options are considered as the only possible valid alignments, and the most frequently occurring alignment (highest r) is chosen. These frequencies are dynamically updated after successfully aligning a pair. For exceptional situations, where there is no character in the target string to match with the source character s_i, it is aligned with the empty string.

It is possible that none of the four valid alignment options have occurred previously (that is, $r = 0$ for each). This situation can arise in two ways: first, such a tuple may simply not have occurred in the training data; and, second, the previous alignment in the current string pair may have been incorrect. To account for this second possibility, a partial backtracking is considered. Most misalignments are derived from the simultaneous comparison of alignment possibilities, giving the highest priority to the most frequent. For example if S=bbc, T=بس and $\mathcal{A} = \{$(b,ب,100),(bb,ب,40),(c,س,60)$\}$, starting from the initial position s_1 and t_1, the first alignment choice is (b,ب,101). However immediately after, we face the problem of aligning the second "b". There are two solutions: inserting ε and adding the triple (b,ε,1), or backtracking the previous alignment and substituting that with the less frequent but possible alignment of (bb,ب,41). The second solution is a better choice as it adds less ambiguous alignments containing ε. At the end, the alignment set is updated as $\mathcal{A} = \{$(b,ب,100),(bb,ب,41),(c,س,61)$\}$.

In case of equal frequencies, we check possible subsequent alignments to decide on which alignment should be chosen. For example, if (b,ب,100) and (bb,ب,100) both exist as possible options, we consider if choosing the former leads to a subsequent ε insertion. If so, we opt for the latter.

At the end of a string, if just one character in the target string remains unaligned while the last alignment is a ε insertion, that final alignment will be substituted for ε. This usually happens when the alignment of final characters is not yet registered in the alignment set, mainly because Persian speakers tend to transliterate the final vowels to consonants to preserve their existence in the word. For example, in the word "Jose" the final "e" might be transliterated to "ه" which is a consonant ("h") and therefore is not captured in *Step 1*.

Backparsing

The process of aligning words explained above can handle words with already known components in the alignment set $\mathcal{A}$ (the frequency of occurrence is greater than zero). However, when this is not the case, the system may repeatedly insert ε while part or all of the target characters are left intact (unsuccessful alignment). In such cases, processing the source and target backwards helps to find the problematic substrings: *backparsing*.

The poorly aligned substrings of the source and target are taken as new pairs of strings, which are then reintroduced into the system as new entries. Note that they themselves are not subject to backparsing. Most strings of repeating nulls can be broken up this way, and in the worst case will remain as one tuple in the alignment set.

To clarify, consider the example given in Figure 1. For the word pair (patricia,پتریشیا), where an association between "c" and "ش" is not yet registered. Forward parsing, as shown in the figure, does not resolve all target characters; after the incorrect alignment of "c" with "ε", subsequent characters are also aligned with null, and the substring "شیا" remains intact. Backward parsing, shown in the next line of the figure, is also not successful. It is able to correctly align the last two characters of the string, before generating repeated null alignments. Therefore, the central region — substrings of the source and target which remained unaligned plus one extra aligned segment to the left and right — is entered as a new pair to the system (ici,یشی), as shown in the line labelled *Input 2* in the figure. This new input meets *Step 1* requirements, and is aligned successfully. The resulting tuples are then merged with the alignment set $\mathcal{A}$.

An advantage of our backparsing strategy is that it takes care of casual transliterations happening due to elision and epenthesis (adding or removing extra sounds). It is not only in translation that people may add extra words to make fluent target text; for transliteration also, it is possible that spurious characters are introduced for fluency. However, this often follows patterns, such as adding vowels to the target form. These irregularities are consistently covered in the backparsing strategy, where they remain connected to their previous character.

4 Transliteration Method

Transliteration algorithms use aligned data (the output from the alignment process, a_{e2p} or a_{p2e} alignment tuples) for training to derive transformation rules. These rules are then used to generate a target word T given a new input source word S.

Initial alignment set:	
$\mathcal{A}$ = {(p,پ,42),(a,ا,320),(a,ε,99),(a,ای,10),(a,ی,35),(r,ر,200),(i,ی,60),(i,ε,5),(c,س,80),(c,چ,25),(t,ت,51)}	
Input:	(patricia,پتریشیا) $q_S = \mathcal{CVCVCV}$ $q_T = \mathcal{CVCV}$
Step 1:	$q_S \neq q_T$
Forward alignment:	(p,پ,43), (a,ε,100), (t,ت,52), (r,ر,201), (i,ی,61), <u>(c,ε,1)</u>, (i,ε,6), (a,ε,100)
Backward alignment:	(a,ا,321), (i,ی,61), <u>(c,ε,1)</u>, (i,ε,6), (r,ε,1), (t,ε,1), (a,ε,100), (p,ε,1)
Input 2:	(ici,یشی) $q_S = \mathcal{VCV}$ $q_T = \mathcal{VCV}$
Step 1:	(i,ی,61),(c,ش,1), (i,ی,61)
Final Alignment:	a_{e2p} = ((p,پ),(a,ε),(t,ت),((r,ر),(i,ی),(c,ش),(i,ی),(a,ا))
Updated alignment set:	
$\mathcal{A}$ = {(p,پ,43),(a,ا,321),(a,ε,100),(a,ای,10),(a,ی,35),(r,ر,201),(i,ی,62),(i,ε,5),(c,س,80),(c,چ,25),(c,ش,1),(t,ت,52)}	

Figure 1: A backparsing example. Note middle tuples in forward and backward parsings are not merged in $\mathcal{A}$ till the alignment is successfully completed.

Method	Intermediate Sequence	Segment(Pattern)	Backoff
Bigram	N/A	#<u>s</u>, s<u>h</u>, h<u>e</u>, e<u>l</u>, l<u>l</u>, l<u>e</u>, e<u>y</u>	s,h,e,l,e,y
CV-MODEL1	$CCVCCV$	<u>sh</u>(CC), h<u>e</u>l(CVC), <u>ll</u>(CC), ll<u>ey</u>($\mathcal{CV}$)	s(C), h(C), e($\mathcal{V}$), l(C), e($\mathcal{V}$), y($\mathcal{V}$)
CV-MODEL2	$CCVCCV$	<u>sh</u>(CC), <u>e</u>(CVC), <u>ll</u>(CC), <u>ey</u>($\overline{\mathcal{C}}\mathcal{V}$)	As Above.
CV-MODEL3	$\mathcal{CVCV}$	#<u>sh</u>($\mathcal{C}$), <u>e</u>($\mathcal{CVC}$), <u>ll</u>($\mathcal{C}$), <u>ey</u>($\mathcal{C}\overline{\mathcal{V}}$)	sh($\mathcal{C}$), s($\mathcal{C}$), h($\mathcal{C}$), e($\mathcal{V}$), l($\mathcal{C}$), e($\mathcal{V}$), y($\mathcal{V}$)

Figure 2: An example of transliteration for the word pair (shelley,شلی). Underlined characters are actually transliterated for each segment.

4.1 Baseline

Most transliteration methods reported in the literature — either grapheme- or phoneme-based — use n-grams (AbdulJaleel and Larkey, 2003; Jung et al., 2000). The n-gram-based methods differ mainly in the way that words are segmented, both for training and transliteration generation. A simple n-gram based method works only on single characters (unigram) and transformation rules are defined as $s_i \rightarrow t_j$, while an advanced method may take the surrounding context into account (Jung et al., 2000). We found that using one past symbol (bigram model) works better than other n-gram based methods for English to Persian transliteration (Karimi et al., 2006).

Our collapsed-vowel methods consider language knowledge to improve the string segmentation of n-gram techniques (Karimi et al., 2006). The process begins by generating the *consonant-vowel* sequence (Definition 2) of a source word. For example, the word "shelley" is represented by the sequence $p = CCVCCVV$. Then, following the collapsed vowel concept (Definition 3), this sequence becomes "$CCVCC\mathcal{V}$". These approaches, which we refer to as CV-MODEL1 and CV-MODEL2 respectively, partition these sequences using basic patterns (C and $\mathcal{V}$) and main patterns (CC, $C\mathcal{V}C$, $\mathcal{V}C$ and $C\mathcal{V}$). In the training phase, transliteration rules are formed according to the boundaries of the defined patterns and their aligned counterparts (based on a_{e2p} or a_{p2e}) in the target language word T. Similar segmentation is applied during the transliteration generation stage.

4.2 The Proposed Transliteration Approach

The restriction on the context length of consonants imposed by CV-MODEL1 and CV-MODEL2 makes the transliteration of consecutive consonants mapping to a particular character in the target language difficult. For example, "ght" in English maps to only one character in Persian: "ت". Dealing with languages which have different alphabets, and for which the number of characters in their alphabets also differs (such as 26 and 32 for English and Persian), increases the possibility of facing these cases, especially when moving from the language with smaller alphabet size to the one with a larger size. To more effectively address this, we propose a *collapsed consonant and vowel* method (CV-MODEL3) which uses the full reduced sequence (Definition 3), rather than simply reduced vowel sequences. Although recognition of consonant segments is based on the vowel positions, consonants are considered as independent blocks in each string. Conversely, vowels are transliterated in the context of surrounding consonants, as demonstrated in the example below.

A special symbol is used to indicate the start and/or end of each word if the beginning and end of the word is a consonant respectively. Therefore, for the words starting or ending with consonants, the symbol "#" is added, which is treated as a consonant and therefore grouped in the consonant segment. An example of applying this technique is shown in Figure 2 for the string "shelley". In this example, "sh" and "ll" are treated as two consonant segments, where the transliteration of individual characters inside a segment is dependent on the other members but not the surrounding segments. However, this is not the case for vowel sequences which incorporate a level of knowledge about any segment neighbours. Therefore, for the example "shelley", the first segment is "sh" which belongs to $\mathcal{C}$ pattern. During transliteration, if "#sh" does not appear in any existing rules, a backoff splits the segment to smaller segments: "#" and "sh", or "s"and "h". The second segment contains the vowel "e". Since this vowel is surrounded by consonants, the segment pattern is $\mathcal{CVC}$. In this case, backoff only applies for vowels as consonants are supposed to be part of their own independent segments. That is, if search in the rules of pattern $\mathcal{CVC}$ was unsuccessful, it looks for "e" in $\mathcal{V}$ pattern. Similarly, segmentation for this word continues with "ll" in $\mathcal{C}$ pattern and "ey" in $\mathcal{CV}$ pattern ("y" is an *approximant*, and therefore considered as a vowel when transliterating English to Persian).

4.3 Rules for Back-Transliteration

Written Persian ignores short vowels, and only long vowels appear in text. This causes most English vowels to disappear when transliterating from English to Persian; hence, these vowels must be restored during back-transliteration.

When the initial transliteration happens from English to Persian, the transliterator (whether human or machine) uses the rules of transliterating from English as the source language. Therefore, transliterating back to the original language should consider the original process, to avoid losing essential information. In terms of segmentation in *collapsed-vowel* models, different patterns define segment boundaries in which vowels are necessary clues. Although we do not have most of these vowels in the transliteration generation phase, it is possible to benefit from their existence in the training phase. For example, using CV-MODEL3, the pair (مرکل,merkel) with q_S=$\mathcal{C}$ and a_{p2e}=((م,me),(ر,r),(ک,ke),(ل,l)), produces just one transformation rule "مرکل → merkel" based on a $\mathcal{C}$ pattern. That is, the Persian string contains no vowel characters. If, during the transliteration generation phase, a source word "مرکل" (S=مرکل) is entered, there would be one and only one output of "merkel", while an alternative such as "mercle" might be required instead. To avoid overfitting the system by long consonant clusters, we perform segmentation based on the English q sequence, but categorise the rules based on their Persian segment counterparts. That is, for the pair (مرکل,merkel) with a_{e2p}=((m,م),(e,ε),(r,ر),(k,ک),(e,ε),(l,ل)), these rules are generated (with category patterns given in parenthesis): م → m ($\mathcal{C}$), رک → rk ($\mathcal{C}$), ل → l ($\mathcal{C}$), مرک → merk ($\mathcal{C}$), رکل → rkel ($\mathcal{C}$). We call the suggested training approach *reverse segmentation*.

Reverse segmentation avoids clustering all the consonants in one rule, since many English words might be transliterated to all-consonant Persian words.

4.4 Transliteration Generation and Ranking

In the transliteration generation stage, the source word is segmented following the same process of segmenting words in training stage, and a probability is computed for each generated target word:

$$P(T|S) = \prod_{k=1}^{|K|} P(\hat{T}_k|\hat{S}_k),$$

where $|K|$ is the number of distinct source segments. $P(\hat{T}_k|\hat{S}_k)$ is the probability of the $\hat{S}_k \rightarrow \hat{T}_k$ transformation rule, as obtained from the training stage:

$$P(\hat{T}_k|\hat{S}_k) = \frac{\text{frequency of } \hat{S}_k \rightarrow \hat{T}_k}{\text{frequency of } \hat{S}_k},$$

where frequency of $\hat{S}_k$ is the number of its occurrence in the transformation rules. We apply a tree structure, following Dijkstra's α-shortest path, to generate the α highest scoring (most probable) transliterations, ranked based on their probabilities.

Corpus		Baseline			CV-MODEL3	
		Bigram	CV-MODEL1	CV-MODEL2	GIZA++	New Alignment
Small Corpus						
	TOP-1	58.0 (2.2)	61.7 (3.0)	60.0 (3.9)	67.4 (5.5)	72.2 (2.2)
	TOP-5	85.6 (3.4)	80.9 (2.2)	86.0 (2.8)	90.9 (2.1)	92.9 (1.6)
	TOP-10	89.4 (2.9)	82.0 (2.1)	91.2 (2.5)	93.8 (2.1)	93.5 (1.7)
Large Corpus						
	TOP-1	47.2 (1.0)	50.6 (2.5)	47.4 (1.0)	55.3 (0.8)	59.8 (1.1)
	TOP-5	77.6 (1.4)	79.8 (3.4)	79.2 (1.0)	84.5 (0.7)	85.4 (0.8)
	TOP-10	83.3 (1.5)	84.9 (3.1)	87.0 (0.9)	89.5 (0.4)	92.6 (0.7)

Table 1: Mean (standard deviation) word accuracy (%) for English to Persian transliteration.

5 Experiments

To investigate the effectiveness of CV-MODEL3 and the new alignment approach on transliteration, we first compare CV-MODEL3 with baseline systems, employing GIZA++ for alignment generation during system training. We then evaluate the same systems, using our new alignment approach. Back-transliteration is also investigated, applying both alignment systems and *reverse segmentation*. In all our experiments, we used ten-fold cross-validation. The statistical significance of different performance levels are evaluated using a paired t-test. The notation TOP-X indicates the first X transliterations prodcued by the automatic methods.

We used two corpora of word pairs in English and Persian: the first, called *Large*, contains 16,670 word pairs; the second, *Small*, contains 1,857 word pairs, and are described fully in our previous paper (Karimi et al., 2006).

The results of transliteration experiments are evaluated using word accuracy (Kang and Choi, 2000) which measures the proportion of transliterations that are correct out of the test corpus.

5.1 Accuracy of Transliteration Approaches

The results of our experiments for transliterating English to Persian, using GIZA++ for alignment generation, are shown in Table 1. CV-MODEL3 outperforms all three baseline systems significantly in TOP-1 and TOP-5 results, for both Persian corpora. TOP-1 results were improved by 9.2% to 16.2% (p<0.0001, paired t-test) relative to the baseline systems for the *Small* corpus. For the *Large* corpus, CV-MODEL3 was 9.3% to 17.2% (p<0.0001) more accurate relative to the baseline systems.

The results of applying our new alignment algorithm are presented in the last column of Table 1, comparing word accuracy of CV-MODEL3 using GIZA++ and the new alignment for English to Persian transliteration. Transliteration accuracy increases in TOP-1 for both corpora (a relative increase of 7.1% (p=0.002) for the *Small* corpus and 8.1% (p<0.0001) for the *Large* corpus). The TOP-10 results of the Large corpus again show a relative increase of 3.5% (p=0.004). Although the new alignment also increases the performance for TOP-5 and TOP-10 of the *Small* corpus, these increases are not statistically significant.

5.2 Accuracy of Back-Transliteration

The results of back-transliteration are shown in Table 2. We first consider performance improvements gained from using CV-MODEL3: CV-MODEL3 using GIZA++ outperforms Bigram, CV-MODEL1 and CV-MODEL2 by 12.8% to 40.7% (p<0.0001) in TOP-1 for the *Small* corpus. The corresponding improvement for the *Large* corpus is 12.8% to 74.2% (p<0.0001).

The fifth column of the table shows the performance increase when using CV-MODEL3 with the new alignment algorithm: for the *Large* corpus, the new alignment approach gives a relative increase in accuracy of 15.5% for TOP-5 (p<0.0001) and 10% for TOP-10 (p=0.005). The new alignment method does not show a significant difference using CV-MODEL3 for the *Small* corpus.

The final column of Table 2 shows the performance of the CV-MODEL3 with the new reverse segmentation approach. Reverse segmentation leads to a significant improvement over the new alignment approach in TOP-1 results for the *Small* corpus by 40.1% (p<0.0001), and 49.4% (p<0.0001) for the *Large* corpus.

Corpus		Bigram	CV-MODEL1	CV-MODEL2	CV-MODEL3		
					GIZA++	New Alignment	Reverse
Small Corpus							
	TOP-1	23.1 (2.0)	28.8 (4.6)	24.9 (2.8)	32.5 (3.6)	34.4 (3.8)	48.2 (2.9)
	TOP-5	40.8 (3.1)	51.0 (4.8)	52.9 (3.4)	56.0 (3.5)	54.8 (3.7)	68.1 (4.9)
	TOP-10	50.1 (4.1)	58.2 (5.3)	63.2 (3.1)	64.2 (3.2)	63.8 (3.6)	75.7 (4.2)
Large Corpus							
	TOP-1	10.1 (0.6)	15.6 (1.0)	12.0 (1.0)	17.6 (0.8)	18.0 (1.2)	26.9 (0.7)
	TOP-5	20.6 (1.2)	31.7 (0.9)	28.0 (0.7)	36.2 (0.5)	41.8 (1.2)	41.3 (1.7)
	TOP-10	27.2 (1.0)	40.1 (1.1)	37.4 (0.8)	46.0 (0.8)	50.6 (1.1)	49.3 (1.6)

Table 2: Comparison of mean (standard deviation) word accuracy (%) for Persian to English transliteration.

6 Conclusions

We have presented a new algorithm for English to Persian transliteration, and a novel alignment algorithm applicable for transliteration. Our new transliteration method (CV-MODEL3) outperforms the previous approaches for English to Persian, increasing word accuracy by a relative 9.2% to 17.2% (TOP-1), when using GIZA++ for alignment in training. This method shows further 7.1% to 8.1% increase in word accuracy (TOP-1) with our new alignment algorithm.

Persian to English back-transliteration is also investigated, with CV-MODEL3 significantly outperforming other methods. Enriching this model with a new reverse segmentation algorithm gives rise to further accuracy gains in comparison to directly applying English to Persian methods.

In future work we will investigate whether phonetic information can help refine our CV-MODEL3, and experiment with manually constructed rules as a baseline system.

Acknowledgments

This work was supported in part by the Australian government IPRS program (SK) and an ARC Discovery Project Grant (AT).

References

Nasreen AbdulJaleel and Leah S. Larkey. 2003. Statistical transliteration for English-Arabic cross language information retrieval. In *Conference on Information and Knowledge Management*, pages 139–146.

Slaven Bilac and Hozumi Tanaka. 2005. Direct combination of spelling and pronunciation information for robust back-transliteration. In *Conferences on Computational Linguistics and Intelligent Text Processing*, pages 413–424.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computional Linguistics*, 19(2):263–311.

Michael A. Covington. 1996. An algorithm to align words for historical comparison. *Computational Linguistics*, 22(4):481–496.

Wei Gao, Kam-Fai Wong, and Wai Lam. 2004. Improving transliteration with precise alignment of phoneme chunks and using contextual features. In *Asia Information Retrieval Symposium*, pages 106–117.

Sung Young Jung, Sung Lim Hong, and Eunok Paek. 2000. An English to Korean transliteration model of extended Markov window. In *Conference on Computational Linguistics*, pages 383–389.

Byung-Ju Kang and Key-Sun Choi. 2000. Automatic transliteration and back-transliteration by decision tree learning. In *Conference on Language Resources and Evaluation*, pages 1135–1411.

Sarvnaz Karimi, Andrew Turpin, and Falk Scholer. 2006. English to Persian transliteration. In *String Processing and Information Retrieval*, pages 255–266.

Alexandre Klementiev and Dan Roth. 2006. Weakly supervised named entity transliteration and discovery from multilingual comparable corpora. In *Association for Computational Linguistics*, pages 817–824.

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

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

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Jong-Hoon Oh and Key-Sun Choi. 2002. An English-Korean transliteration model using pronunciation and contextual rules. In *Conference on Computational Linguistics*.

Paola Virga and Sanjeev Khudanpur. 2003. Transliteration of proper names in cross-language applications. In *ACM SIGIR Conference on Research and Development on Information Retrieval*, pages 365–366.

Dmitry Zelenko and Chinatsu Aone. 2006. Discriminative methods for transliteration. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing.*, pages 612–617.