Hindi-to-Urdu Machine Translation Through Transliteration

Nadir Durrani **Hassan Sajjad** **Alexander Fraser** **Helmut Schmid**
Institute for Natural Language Processing
University of Stuttgart
{durrani,sajjad,fraser,schmid}@ims.uni-stuttgart.de

Abstract

We present a novel approach to integrate transliteration into Hindi-to-Urdu statistical machine translation. We propose two probabilistic models, based on conditional and joint probability formulations, that are novel solutions to the problem. Our models consider both transliteration and translation when translating a particular Hindi word given the context whereas in previous work transliteration is only used for translating OOV (out-of-vocabulary) words. We use transliteration as a tool for disambiguation of Hindi homonyms which can be both translated or transliterated or transliterated differently based on different contexts. We obtain final BLEU scores of 19.35 (conditional probability model) and 19.00 (joint probability model) as compared to 14.30 for a baseline phrase-based system and 16.25 for a system which transliterates OOV words in the baseline system. This indicates that transliteration is useful for more than only translating OOV words for language pairs like Hindi-Urdu.

1 Introduction

Hindi is an official language of India and is written in Devanagari script. Urdu is the national language of Pakistan, and also one of the state languages in India, and is written in Perso-Arabic script. Hindi inherits its vocabulary from Sanskrit while Urdu descends from several languages including Arabic, Farsi (Persian), Turkish and Sanskrit. Hindi and Urdu share grammatical structure and a large proportion of vocabulary that they both inherited from Sanskrit. Most of the verbs and closed-class words (pronouns, auxiliaries, case-markers, etc) are the same. Because both languages have lived together for centuries, some Urdu words which originally came from Arabic and Farsi have also mixed into Hindi and are now part of the Hindi vocabulary. The spoken form of the two languages is very similar.

The extent of overlap between Hindi and Urdu vocabulary depends upon the domain of the text. Text coming from the literary domain like novels or history tend to have more Sanskrit (for Hindi) and Persian/Arabic (for Urdu) vocabulary. However, news wire that contains text related to media, sports and politics, etc., is more likely to have common vocabulary.

In an initial study on a small news corpus of 5000 words, randomly selected from BBC[1] News, we found that approximately 62% of the Hindi types are also part of Urdu vocabulary and thus can be transliterated while only 38% have to be translated. This provides a strong motivation to implement an end-to-end translation system which strongly relies on high quality transliteration from Hindi to Urdu.

Hindi and Urdu have similar sound systems but transliteration from Hindi to Urdu is still very hard because some phonemes in Hindi have several orthographic equivalents in Urdu. For example the "z" sound[2] can only be written as ज़ whenever it occurs in a Hindi word but can be written as ذ, ز, ض and ظ in an Urdu word. Transliteration becomes non-trivial in cases where the multiple orthographic equivalents for a Hindi word are all valid Urdu words. Context is required to resolve ambiguity in such cases. Our transliterator (described in sections 3.1.2 and 4.1.3) gives an accuracy of 81.6% and a 25-best accuracy of 92.3%.

Transliteration has been previously used only as a back-off measure to translate NEs (Name Entities) and OOV words in a pre- or post-processing step. The problem we are solving is more difficult than techniques aimed at handling OOV words,

[1] http://www.bbc.co.uk/hindi/index.shtml
[2] All sounds are represented using SAMPA notation.

Hindi	Urdu	SAMPA	Gloss
आम	عام / آم	Am	Mango/Ordinary
जाली	جالی / جعلی	d_ZAli	Fake/Net
शेर	شعر / شیر	Ser	Lion/Verse

Table 1: Hindi Words That Can Be Transliterated Differently in Different Contexts

Hindi	Urdu	SAMPA	Gloss
सीमा	سیما / سرحد	simA	Border/Seema
अंबर	امبر / آسمان	Amb@r	Sky/Ambar
विजय	وِجے / جیت	vId_Ze	Victory/Vijay

Table 2: Hindi Words That Can Be Translated or Transliterated in Different Contexts

which focus primarily on name transliteration, because we need different transliterations in different contexts; in their case context is irrelevant. For example: consider the problem of transliterating the English word "read" to a phoneme representation in the context "I will read" versus the context "I have read". An example of this for Hindi to Urdu transliteration: the two Urdu words صورت (face/condition) and سورت (chapter of the Koran) are both written as सूरत (sur@t_d) in Hindi. The two are pronounced identically in Urdu but written differently. In such cases we hope to choose the correct transliteration by using context. Some other examples are shown in Table 1.

Sometimes there is also an ambiguity of whether to translate or transliterate a particular word. The Hindi word शान्ती, for example, will be translated to سکون (peace, s@kun) when it is a common noun but transliterated to شانتی (Shanti, SAnt_di) when it is a proper name. We try to model whether to translate or transliterate in a given situation. Some other examples are shown in Table 2.

The remainder of this paper is organized as follows. Section 2 provides a review of previous work. Section 3 introduces two probabilistic models for integrating translations and transliterations into a translation model which are based on conditional and joint probability distributions. Section 4 discusses the training data, parameter optimization and the initial set of experiments that compare our two models with a baseline Hindi-Urdu phrase-based system and with two transliteration-aided phrase-based systems in terms of BLEU scores (Papineni et al., 2001). Section 5 performs an error analysis showing interesting weaknesses in the initial formulations. We remedy the problems by adding some heuristics and modifications to our models which show improvements in the results as discussed in section 6. Section 7 gives two examples illustrating how our model decides whether to translate or transliterate and how it is able to choose among different valid transliterations given the context. Section 8 concludes the paper.

2 Previous Work

There has been a significant amount of work on transliteration. We can break down previous work into three groups. The first group is generic transliteration work, which is evaluated outside of the context of translation. This work uses either grapheme or phoneme based models to transliterate words lists (Knight and Graehl, 1998; Li et al., 2004; Ekbal et al., 2006; Malik et al., 2008). The work by Malik et al. addresses Hindi to Urdu transliteration using hand-crafted rules and a phonemic representation; it ignores translation context.

A second group deals with out-of-vocabulary words for SMT systems built on large parallel corpora, and therefore focuses on name transliteration, which is largely independent of context. Al-Onaizan and Knight (2002) transliterate Arabic NEs into English and score them against their respective translations using a modified IBM Model 1. The options are further re-ranked based on different measures such as web counts and using coreference to resolve ambiguity. These re-ranking methodologies can not be performed in SMT at the decoding time. An efficient way to compute and re-rank the transliterations of NEs and integrate them on the fly might be possible. However, this is not practical in our case as our model considers transliterations of all input words and not just NEs. A log-linear block transliteration model is applied to OOV NEs in Arabic to English SMT by Zhao et al. (2007). This work is also transliterating only NEs and not doing any disambiguation. The best method proposed by Kashani et al. (2007) integrates translations provided by external sources such as transliteration or rule-base translation of numbers and dates, for an arbitrary number of entries within the input text. Our work is different from Kashani et al. (2007) in that our model compares transliterations with translations on the fly whereas transliterations in Kashani et al. do not compete with internal phrase tables. They only compete amongst themselves during a second pass of decoding. Hermjakob et al. (2008) use a tagger to identify good candidates for transliteration (which are mostly NEs) in input text and add transliterations to the SMT phrase table dynamically such that they can directly compete with translations during decoding. This is closer to our approach except that we use transliteration as an alternative to translation for all Hindi words. Our focus is disambiguation of Hindi homonyms whereas they are concentrating only on transliterating NE's. Moreover, they are working with a large bitext so they can rely on their translation model and only need to transliterate NEs and OOVs. Our translation model is based on data which is both sparse and noisy. Therefore we pit transliterations against translations for every input word. Sinha (2009) presents a rule-based MT system that uses Hindi as a pivot to translate from English to Urdu. This work also uses transliteration only for the translation of unknown words. Their work can not be used for direct translation from Hindi to Urdu (independently of English) "due to various ambiguous mappings that have to be resolved".

The third group uses transliteration models inside of a cross-lingual IR system (AbdulJaleel and Larkey, 2003; Virga and Khudanpur, 2003; Pirkola et al., 2003). Picking a single best transliteration or translation in context is not important in an IR system. Instead, all the options are used by giving them weights and context is typically not taken into account.

3 Our Approach

Both of our models combine a character-based transliteration model with a word-based translation model. Our models look for the most probable Urdu token sequence u_1^n for a given Hindi token sequence h_1^n. We assume that each Hindi token is mapped to exactly one Urdu token and that there is no reordering. The assumption of no reordering is reasonable given the fact that Hindi and Urdu have identical grammar structure and the same word order. An Urdu token might consist of more than one Urdu word[3]. The following sections give a mathematical formulation of our two models, Model-1 and Model-2.

3.1 Model-1 : Conditional Probability Model

Applying a noisy channel model to compute the most probable translation $\hat{u}_1^n$, we get:

$$\arg\max_{u_1^n} p(u_1^n|h_1^n) = \arg\max_{u_1^n} p(u_1^n)p(h_1^n|u_1^n) \tag{1}$$

3.1.1 Language Model

The **language model (LM)** $p(u_1^n)$ is implemented as an n-gram model using the SRILM-Toolkit (Stolcke, 2002) with Kneser-Ney smoothing. The parameters of the language model are learned from a monolingual Urdu corpus. The language model is defined as:

$$p(u_1^n) = \prod_{i=1}^{n} p_{LM}(u_i|u_{i-k}^{i-1}) \tag{2}$$

where k is a parameter indicating the amount of context used (e.g., $k = 4$ means 5-gram model). u_i can be a single or a multi-word token. A multi-word token consists of two or more Urdu words. For a multi-word u_i we do multiple language model look-ups, one for each u_{i_x} in $u_i = u_{i_1}, \ldots, u_{i_m}$ and take their product to obtain the value $p_{LM}(u_i|u_{i-k}^{i-1})$.

Language Model for Unknown Words: Our model generates transliterations that can be known or unknown to the language model and the translation model. We refer to the words known to the language model and to the translation model as LM-known and TM-known words respectively and to words that are unknown as LM-unknown and TM-unknown respectively.

We assign a special value ψ to the LM-unknown words. If one or more u_{i_x} in a multi-word u_i are LM-unknown we assign a language model score $p_{LM}(u_i|u_{i-k}^{i-1}) = \psi$ for the entire u_i, meaning that we consider partially known transliterations to be as bad as fully unknown transliterations. The parameter ψ controls the trade-off between LM-known and LM-unknown transliterations. It does not influence translation options because they are always LM-known in our case. This is because our monolingual corpus also contains the Urdu part of translation corpus. The optimization of ψ is described in section 4.2.1.

[3]This occurs frequently in case markers with nouns, derivational affixes and compounds etc. These are written as single words in Hindi as opposed to Urdu where they are written as two words. For example खूबसूरत (beautiful ; xubsur@t_d) and आपका (your's ; ApkA) are written as خوب صورت and آپ کا respectively in Urdu.

3.1.2 Translation Model

The **translation model (TM)** $p(h_1^n|u_1^n)$ is approximated with a context-independent model:

$$p(h_1^n|u_1^n) = \prod_{i=1}^{n} p(h_i|u_i) \quad (3)$$

where h_i and u_i are Hindi and Urdu tokens respectively. Our model estimates the conditional probability $p(h_i|u_i)$ by interpolating a word-based model and a character-based (transliteration) model.

$$p(h_i|u_i) = \lambda p_w(h_i|u_i) + (1-\lambda)p_c(h_i|u_i) \quad (4)$$

The parameters of the **word-based translation model** $p_w(h|u)$ are estimated from the word alignments of a small parallel corpus. We only retain 1-1/1-N (1 Hindi word, 1 or more Urdu words) alignments and throw away N-1 and M-N alignments for our models. This is further discussed in section 4.1.1.

The **character-based transliteration model** $p_c(h|u)$ is computed in terms of $p_c(h,u)$, a joint character model, which is also used for Chinese-English back-transliteration (Li et al., 2004) and Bengali-English name transliteration (Ekbal et al., 2006). The character-based transliteration probability is defined as follows:

$$\begin{aligned} p_c(h,u) &= \sum_{a_1^n \in align(h,u)} p(a_1^n) \\ &= \sum_{a_1^n \in align(h,u)} \prod_{i=1}^{n} p(a_i|a_{i-k}^{i-1}) \quad (5) \end{aligned}$$

where a_i is a pair consisting of the i-th Hindi character h_i and the sequence of 0 or more Urdu characters that it is aligned with. A sample alignment is shown in Table 3(b) in section 4.1.3. Our best results are obtained with a 5-gram model. The parameters $p(a_i|a_{i-k}^{i-1})$ are estimated from a small transliteration corpus which we automatically extracted from the translation corpus. The extraction details are also discussed in section 4.1.3. Because our overall model is a conditional probability model, joint-probabilities are marginalized using character-based prior probabilities:

$$p_c(h|u) = \frac{p_c(h,u)}{p_c(u)} \quad (6)$$

The **prior probability** $p_c(u)$ of the character sequence $u = c_1^m$ is defined with a character-based language model:

$$p_c(u) = \prod_{i=1}^{m} p(c_i|c_{i-k}^{i-1}) \quad (7)$$

The parameters $p(c_i|c_{i-k}^{i-1})$ are estimated from the Urdu part of the character-aligned transliteration corpus. Replacing (6) in (4) we get:

$$p(h_i|u_i) = \lambda p_w(h_i|u_i) + (1-\lambda)\frac{p_c(h_i,u_i)}{p_c(u_i)} \quad (8)$$

Having all the components of our model defined we insert (8) and (2) in (1) to obtain the final equation:

$$\begin{aligned} \hat{u}_1^n = \arg\max_{u_1^n} \prod_{i=1}^{n} p_{LM}(u_i|u_{i-k}^{i-1})[\lambda p_w(h_i|u_i) \\ + (1-\lambda)\frac{p_c(h_i,u_i)}{p_c(u_i)}] \quad (9) \end{aligned}$$

The optimization of the interpolating factor λ is discussed in section 4.2.1.

3.2 Model-2 : Joint Probability Model

This section briefly defines a variant of our model where we interpolate joint probabilities instead of conditional probabilities. Again, the **translation model** $p(h_1^n|u_1^n)$ is approximated with a context-independent model:

$$p(h_1^n|u_1^n) = \prod_{i=1}^{n} p(h_i|u_i) = \prod_{i=1}^{n} \frac{p(h_i,u_i)}{p(u_i)} \quad (10)$$

The joint probability $p(h_i,u_i)$ of a Hindi and an Urdu word is estimated by interpolating a word-based model and a character-based model.

$$p(h_i,u_i) = \lambda p_w(h_i,u_i) + (1-\lambda)p_c(h_i,u_i) \quad (11)$$

and the prior probability $p(u_i)$ is estimated as:

$$p(u_i) = \lambda p_w(u_i) + (1-\lambda)p_c(u_i) \quad (12)$$

The parameters of the translation model $p_w(h_i,u_i)$ and the word-based prior probabilities $p_w(u_i)$ are estimated from the 1-1/1-N word-aligned corpus (the one that we also used to estimate translation probabilities $p_w(h_i|u_i)$ previously).

The character-based transliteration probability $p_c(h_i,u_i)$ and the character-based prior probability $p_c(u_i)$ are defined by (5) and (7) respectively in the previous section. Putting (11) and (12) in (10) we get

$$p(h_1^n|u_1^n) = \prod_{i=1}^{n} \frac{\lambda p_w(h_i,u_i) + (1-\lambda)p_c(h_i,u_i)}{\lambda p_w(u_i) + (1-\lambda)p_c(u_i)} \tag{13}$$

The idea is to interpolate joint probabilities and divide them by the interpolated marginals. The final equation for Model-2 is given as:

$$\hat{u}_1^n = \arg\max_{u_1^n} \prod_{i=1}^{n} p_{LM}(u_i|u_{i-k}^{i-1}) \times \frac{\lambda p_w(h_i,u_i) + (1-\lambda)p_c(h_i,u_i)}{\lambda p_w(u_i) + (1-\lambda)p_c(u_i)} \tag{14}$$

3.3 Search

The decoder performs a stack-based search using a beam-search algorithm similar to the one used in Pharoah (Koehn, 2004a). It searches for an Urdu string that maximizes the product of translation probability and the language model probability (equation 1) by translating one Hindi word at a time. It is implemented as a two-level process. At the lower level, it computes n-best transliterations for each Hindi word h_i according to $p_c(h,u)$. The joint probabilities given by $p_c(h,u)$ are marginalized for each Urdu transliteration to give $p_c(h|u)$. At the higher level, transliteration probabilities are interpolated with $p_w(h|u)$ and then multiplied with language model probabilities to give the probability of a hypothesis. We use 20-best translations and 25-best transliterations for $p_w(h|u)$ and $p_c(h|u)$ respectively and a 5-gram language model.

To keep the search space manageable and time complexity polynomial we apply pruning and recombination. Since our model uses monotonic decoding we only need to recombine hypotheses that have the same context (last n-1 words). Next we do histogram-based pruning, maintaining the 100-best hypotheses for each stack.

4 Evaluation

4.1 Training

This section discusses the training of the different model components.

4.1.1 Translation Corpus

We used the freely available EMILLE Corpus as our bilingual resource which contains roughly 13,000 Urdu and 12,300 Hindi sentences. From these we were able to sentence-align 7000 sentence pairs using the sentence alignment algorithm given by Moore (2002).

The word alignments for this task were extracted by using GIZA++ (Och and Ney, 2003) in both directions. We extracted a total of 107323 alignment pairs (5743 N-1 alignments, 8404 M-N alignments and 93176 1-1/1-N alignments). Of these alignments M-N and N-1 alignment pairs were ignored. We manually inspected a sample of 1000 instances of M-N/N-1 alignments and found that more than 70% of these were (totally or partially) wrong. Of the 30% correct alignments, roughly one-third constitute N-1 alignments. Most of these are cases where the Urdu part of the alignment actually consists of two (or three) words but was written without space because of lack of standard writing convention in Urdu. For example جا سکتے (can go ; d_ZA s@kt_de) is alternatively written as جاسکتے (can go ; d_ZAs@kt_de) i.e. without space. We learned that these N-1 translations could be safely dropped because we can generate a separate Urdu word for each Hindi word. For valid M-N alignments we observed that these could be broken into 1-1/1-N alignments in most of the cases. We also observed that we usually have coverage of the resulting 1-1 and 1-N alignments in our translation corpus. Looking at the noise in the incorrect alignments we decided to drop N-1 and M-N cases. We do not model deletions and insertions so we ignored null alignments. Also 1-N alignments with gaps were ignored. Only the alignments with contiguous words were kept.

4.1.2 Monolingual Corpus

Our monolingual Urdu corpus consists of roughly 114K sentences. This comprises 108K sentences from the data made available by the University of Leipzig[4] + 5600 sentences from the training data of each fold during cross validation.

4.1.3 Transliteration Corpus

The training corpus for transliteration is extracted from the 1-1/1-N word-alignments of the EMILLE corpus discussed in section 4.1.1. We use an edit distance algorithm to align this training corpus at the character level and we eliminate translation pairs with high edit distance which are unlikely to be transliterations.

[4]http://corpora.informatik.uni-leipzig.de/

We used our knowledge of the Hindi and Urdu scripts to define the initial character mapping. The mapping was further extended by looking into available Hindi-Urdu transliteration systems[5,6] and other resources (Gupta, 2004; Malik et al., 2008; Jawaid and Ahmed, 2009). Each pair in the character map is assigned a cost. A Hindi character that always map to only one Urdu character is assigned a cost of 0 whereas the Hindi characters that map to different Urdu characters are assigned a cost of 0.2. The edit distance metric allows insert, delete and replace operations. The hand-crafted pairs define the cost of replace operations. We set a cost of 0.6 for deletions and insertions. These costs were optimized on held out data. The details of optimization are not mentioned due to limited space. Using this metric we filter out the word pairs with high edit-distance to extract our transliteration corpus. We were able to extract roughly 2100 unique pairs along with their alignments. The resulting alignments are modified by merging unaligned $\emptyset \rightarrow 1$ (no character on source side, 1 character on target side) or $\emptyset \rightarrow N$ alignments with the preceding alignment pair. If there is no preceding alignment pair then it is merged with the following pair. Table 3 gives an example showing initial alignment (a) and the final alignment (b) after applying the merge operation. Our model retains $1 \rightarrow \emptyset$ and $N \rightarrow \emptyset$ alignments as deletion operations.

a)	Hindi	$\emptyset$	b	c	$\emptyset$	e	f
	Urdu	A	XY	C	D	$\emptyset$	F
b)	Hindi		b	c		e	f
	Urdu		AXY	CD		$\emptyset$	F

Table 3: Alignment (a) Before (b) After Merge

The parameters $p_c(h, u)$ and $p_c(u)$ are trained on the aligned corpus using the SRILM toolkit. We use Add-1 smoothing for unigrams and Kneser-Ney smoothing for higher n-grams.

4.1.4 Diacritic Removal and Normalization

In Urdu, short vowels are represented with diacritics but these are rarely written in practice. In order to keep the data consistent, all diacritics are removed. This loss of information is not harmful when transliterating/translating from Hindi to Urdu because undiacritized text is equally readable to native speakers as its diacritized counter part. However leaving occasional diacritics in the corpus can worsen the problem of data sparsity by creating spurious ambiguity[7].

There are a few Urdu characters that have multiple equivalent Unicodes. All such forms are normalized to have only one representation[8].

4.2 Experimental Setup

We perform a 5-fold cross validation taking 4/5 of the data as training and 1/5 as test data. Each fold comprises roughly 1400 test sentences and 5600 training sentences.

4.2.1 Parameter Optimization

Our model contains two parameters λ (the interpolating factor between translation and transliteration modules) and ψ (the factor that controls the trade-off between LM-known and LM-unknown transliterations). The interpolating factor λ is initialized, inspired by Written-Bell smoothing, with a value of $\frac{N}{N+B}$[9]. We chose a very low value $1e^{-40}$ for the factor ψ initially, favoring LM-known transliterations very strongly. Both of these parameters are optimized as described below.

Because our training data is very sparse we do not use held-out data for parameter optimization. Instead we optimize these parameters by performing a 2-fold optimization for each of the 5 folds. Each fold is divided into two halves. The parameters λ and ψ are optimized on the first half and the other half is used for testing, then optimization is done on the second half and the first half is used for testing. The optimal value for parameter λ occurs between 0.7-0.84 and for the parameter ψ between $1e^{-5}$ and $1e^{-10}$.

4.2.2 Results

Baseline Pb_0: We ran Moses (Koehn et al., 2007) using Koehn's training scripts[10], doing a 5-fold cross validation with no reordering[11]. For the other parameters we use the default values i.e. 5-gram language model and maximum phrase-length= 6. Again, the language model is imple-

[5]CRULP: http://www.crulp.org/software/langproc.htm

[6]Malerkotla.org: http://translate.malerkotla.co.in

[7]It should be noted though that diacritics play a very important role when transliterating in the reverse direction because these are virtually always written in Hindi as dependent vowels.

[8]www.crulp.org/software/langproc/urdunormalization.htm

[9]N is the number of aligned word pairs (tokens) and B is the number of different aligned word pairs (types).

[10]http://statmt.org/wmt08/baseline.html

[11]Results are worse with reordering enabled.

M	$\mathbf{Pb_0}$	$\mathbf{Pb_1}$	$\mathbf{Pb_2}$	$\mathbf{M_1}$	$\mathbf{M_2}$
BLEU	14.3	16.25	16.13	18.6	17.05

Table 4: Comparing Model-1 and Model-2 with Phrase-based Systems

mented as an n-gram model using the SRILM-Toolkit with Kneser-Ney smoothing. Each fold comprises roughly 1400 test sentences, 5000 in training and 600 in dev[12]. We also used two methods to incorporate transliterations in the phrase-based system:

Post-process Pb_1**:** All the OOV words in the phrase-based output are replaced with their top-candidate transliteration as given by our transliteration system.

Pre-process Pb_2**:** Instead of adding transliterations as a post process we do a second pass by adding the unknown words with their top-candidate transliteration to the training corpus and rerun Koehn's training script with the new training corpus. Table 4 shows results (taking arithmetic average over 5 folds) from Model-1 and Model-2 in comparison with three baselines discussed above.

Both our systems (Model-1 and Model-2) beat the baseline phrase-based system with a BLEU point difference of 4.30 and 2.75 respectively. The transliteration aided phrase-based systems Pb_1 and Pb_2 are closer to our Model-2 results but are way below Model-1 results. The difference of 2.35 BLEU points between M_1 and Pb_1 indicates that transliteration is useful for more than only translating OOV words for language pairs like Hindi-Urdu. Our models choose between translations and transliterations based on context unlike the phrase-based systems Pb_1 and Pb_2 which use transliteration only as a tool to translate OOV words.

5 Error Analysis

Based on preliminary experiments we found three major flaws in our initial formulations. This section discusses each one of them and provides some heuristics and modifications that we employ to try to correct deficiencies we found in the two models described in section 3.1 and 3.2.

5.1 Heuristic-1

A lot of errors occur because our translation model is built on very sparse and noisy data. The motivation for this heuristic is to counter wrong alignments at least in the case of verbs and functional words (which are often transliterations). This heuristic favors translations that also appear in the n-best transliteration list over only-translation and only-transliteration options. We modify the translation model for both the conditional and the joint model by adding another factor which strongly weighs translation+transliteration options by taking the square-root of the product of the translation and transliteration probabilities. Thus modifying equations (8) and (11) in Model-1 and Model-2 we obtain equations (15) and (16) respectively:

$$p(h_i|u_i) = \lambda_1 p_w(h_i|u_i) + \lambda_2 \frac{p_c(h_i,u_i)}{p_c(u_i)} + \lambda_3 \sqrt{p_w(h_i|u_i)\frac{p_c(h_i,u_i)}{p_c(u_i)}} \quad (15)$$

$$p(h_i,u_i) = \lambda_1 p_w(h_i,u_i) + \lambda_2 p_c(h_i,u_i) + \lambda_3 \sqrt{p_w(h_i,u_i)p_c(h_i,u_i)} \quad (16)$$

For the optimization of lambda parameters we hold the value of the translation coefficient λ_1[13] and the transliteration coefficient λ_2 constant (using the optimized values as discussed in section 4.2.1) and optimize λ_3 again using 2-fold optimization on all the folds as described above[14].

5.2 Heuristic-2

When an unknown Hindi word occurs for which all transliteration options are LM-unknown then the best transliteration should be selected. The problem in our original models is that a fixed LM probability ψ is used for LM-unknown transliterations. Hence our model selects the transliteration that has the best $\frac{p_c(h_i,u_i)}{p_c(u_i)}$ score i.e. we maximize $p_c(h_i|u_i)$ instead of $p_c(u_i|h_i)$ (or equivalently $p_c(h_i,u_i)$). The reason is an inconsistency in our models. The language model probability of unknown words is uniform (and equal to ψ) whereas the translation model uses the non-uniform prior probability $p_c(u_i)$ for these words. There is another reason why we can not use the

[12]After having the MERT parameters, we add the 600 dev sentences back into the training corpus, retrain GIZA, and then estimate a new phrase table on all 5600 sentences. We then use the MERT parameters obtained before together with the newer (larger) phrase-table set.

[13]The translation coefficient λ_1 is same as λ used in previous models and the transliteration coefficient $\lambda_2 = 1 - \lambda$

[14]After optimization we normalize the lambdas to make their sum equal to 1.

value ψ in this case. Our transliterator model also produces space inserted words. The value of ψ is very small because of which transliterations that are actually LM-unknown, but are mistakenly broken into constituents that are LM-known, will always be preferred over their counter parts. An example of this is अमेरिका (America) for which two possible transliterations as given by our model are امیرکا (AmerIkA, without space) and امیر کا (AmerI kA, with space). The latter version is LM-known as its constituents are LM-known. Our models always favor the latter version. Space insertion is an important feature of our transliteration model. We want our transliterator to tackle compound words, derivational affixes, case-markers with nouns that are written as one word in Hindi but as two or more words in Urdu. Examples were already shown in section 3's footnote.

We eliminate the inconsistency by using $p_c(u_i)$ as the 0-gram back-off probability distribution in the language model. For an LM-unknown transliterations we now get in Model-1:

$$p(u_i|u_{i-k}^{i-1})[\lambda p_w(h_i|u_i) + (1-\lambda)\frac{p_c(h_i,u_i)}{p_c(u_i)}]$$
$$= p(u_i|u_{i-k}^{i-1})[(1-\lambda)\frac{p_c(h_i,u_i)}{p_c(u_i)}]$$
$$= \prod_{j=0}^{k} \alpha(u_{i-j}^{i-1})p_c(u_i)[(1-\lambda)\frac{p_c(h_i,u_i)}{p_c(u_i)}]$$
$$= \prod_{j=0}^{k} \alpha(u_{i-j}^{i-1})[(1-\lambda)p_c(h_i,u_i)]$$

where $\prod_{j=0}^{k} \alpha(u_{i-j}^{i-1})$ is just the constant that SRILM returns for unknown words. The last line of the calculation shows that we simply drop $p_c(u_i)$ if u_i is LM-unknown and use the constant $\prod_{j=0}^{k} \alpha(u_{i-j}^{i-1})$ instead of ψ. A similar calculation for Model-2 gives $\prod_{j=0}^{k} \alpha(u_{i-j}^{i-1})p_c(h_i,u_i)$.

5.3 Heuristic-3

This heuristic discusses a flaw in Model-2. For transliteration options that are TM-unknown, the $p_w(h,u)$ and $p_w(u)$ factors becomes zero and the translation model probability as given by equation (13) becomes:

$$\frac{(1-\lambda)p_c(h_i,u_i)}{(1-\lambda)p_c(u_i)} = \frac{p_c(h_i,u_i)}{p_c(u_i)}$$

In such cases the λ factor cancels out and no weighting of word translation vs. transliteration occurs anymore. As a result of this, transliterations are sometimes incorrectly favored over their translation alternatives.

In order to remedy this problem we assign a minimal probability β to the word-based prior $p_w(u_i)$ in case of TM-unknown transliterations, which prevents it from ever being zero. Because of this addition the translation model probability for LM-unknown words becomes:

$$\frac{(1-\lambda)p_c(h_i,u_i)}{\lambda\beta + (1-\lambda)p_c(u_i)} \text{ where } \beta = \frac{1}{\text{Urdu Types in TM}}$$

	$\mathbf{H_1}$	$\mathbf{H_2}$	$\mathbf{H_{12}}$
$\mathbf{M_1}$	18.86	18.97	19.35
$\mathbf{M_2}$	17.56	17.85	18.34

Table 5: Applying Heuristics 1 and 2 and their Combinations to Model-1 and Model-2

	$\mathbf{H_3}$	$\mathbf{H_{13}}$	$\mathbf{H_{23}}$	$\mathbf{H_{123}}$
$\mathbf{M_2}$	18.52	18.93	18.55	19.00

Table 6: Applying Heuristic 3 and its Combinations with other Heuristics to Model-2

6 Final Results

This section shows the improvement in BLEU score by applying heuristics and combinations of heuristics in both the models. Tables 5 and 6 show the improvements achieved by using the different heuristics and modifications discussed in section 5. We refer to the results as M_xH_y where x denotes the model number, 1 for the conditional probability model and 2 for the joint probability model and y denotes a heuristic or a combination of heuristics applied to that model[15].

Both heuristics (H_1 and H_2) show improvements over their base models M_1 and M_2. Heuristic-1 shows notable improvement for both models in parts of test data which has high number of common vocabulary words. Using heuristic 2 we were able to properly score LM-unknown transliterations against each other. Using these heuristics together we obtain a gain of 0.75 over M-1 and a gain of 1.29 over M-2.

Heuristic-3 remedies the flaw in M_2 by assigning a special value to the word-based prior $p_w(u_i)$ for TM-unknown words which prevents the cancelation of interpolating parameter λ. M_2 combined with heuristic 3 (M_2H_3) results in a 1.47

[15]For example M_1H_1 refers to the results when heuristic-1 is applied to model-1 whereas M_2H_{12} refers to the results when heuristics 1 and 2 are together applied to model 2.

BLEU point improvement and combined with all the heuristics (M_2H_{123}) gives an overall gain of 1.95 BLEU points and is close to our best results (M_1H_{12}). We also performed significance test by concatenating all the fold results. Both our best systems M_1H_{12} and M_2H_{123} are statistically significant ($p < 0.05$)[16] over all the baselines discussed in section 4.2.2.

One important issue that has not been investigated yet is that BLEU has not yet been shown to have good performance in morphologically rich target languages like Urdu, but there is no metric known to work better. We observed that sometimes on data where the translators preferred to translate rather than doing transliteration our system is penalized by BLEU even though our output string is a valid translation. For other parts of the data where the translators have heavily used transliteration, the system may receive a higher BLEU score. We feel that this is an interesting area of research for automatic metric developers, and that a large scale task of translation to Urdu which would involve a human evaluation campaign would be very interesting.

शेर जंगल का राजा है
شیر جنگل کا راجہ ہے
Ser d_Z@ngl kA rAd_ZA he
“Lion is the king of jungle”
इकबाल का एक खूबसूरत शेर है
اقبال کا ایک خوب صورت شعر ہے
AIqbAl kA Aek xub sur@t_d Ser he
“There is a beautiful verse from Iqbal”

Figure 1: Different Transliterations in Different Contexts

फिर भी वह शान्ती से नहीं रह सकता है
پھر بھی وہ سکون سے نہیں رہ سکتا ہے
p_hIr b_hi vh s@kun se n@hẽh s@kt_dA
“Even then he can’t live peacefully”
ओम शान्ती ओम फराह खान की दूसरी फिल्म है
اوم شانتی اوم فراح خان کی دوسری فلم ہے
Aom SAnt_di Aom frhA xAn ki d_dusri fIl@m he
“Om Shanti Om is Farah Khan’s second film”

Figure 2: Translation or Transliteration

7 Sample Output

This section gives two examples showing how our model ($M1H2$) performs disambiguation. Given below are some test sentences that have Hindi homonyms (underlined in the examples) along with Urdu output given by our system. In the first example (given in Figure 1) Hindi word शेर can be transliterated to شیر (Lion) or شعر (Verse) depending upon the context. Our model correctly identifies which transliteration to choose given the context.

In the second example (shown in Figure 2) Hindi word शान्ती can be translated to سکون (peace, s@kun) when it is a common noun but transliterated to شانتی (Shanti, SAnt_di) when it is a proper name. Our model successfully decides whether to translate or transliterate given the context.

8 Conclusion

We have presented a novel way to integrate transliterations into machine translation. In closely related language pairs such as Hindi-Urdu with a significant amount of vocabulary overlap, transliteration can be very effective in machine translation for more than just translating OOV words. We have addressed two problems. First, transliteration helps overcome the problem of data sparsity and noisy alignments. We are able to generate word translations that are unseen in the translation corpus but known to the language model. Additionally, we can generate novel transliterations (that are LM-Unknown). Second, generating multiple transliterations for homograph Hindi words and using language model context helps us solve the problem of disambiguation. We found that the joint probability model performs almost as well as the conditional probability model but that it was more complex to make it work well.

Acknowledgments

The first two authors were funded by the Higher Education Commission (HEC) of Pakistan. The third author was funded by Deutsche Forschungsgemeinschaft grants SFB 732 and MorphoSynt. The fourth author was funded by Deutsche Forschungsgemeinschaft grant SFB 732.

[16] We used Kevin Gimpel’s tester (http://www.ark.cs.cmu.edu/MT/) which uses bootstrap resampling (Koehn, 2004b), with 1000 samples.

References

Nasreen AbdulJaleel and Leah S. Larkey. 2003. Statistical transliteration for English-Arabic cross language information retrieval. In *CIKM 03: Proceedings of the twelfth international conference on Information and knowledge management*, pages 139–146.

Yaser Al-Onaizan and Kevin Knight. 2002. Translating named entities using monolingual and bilingual resources. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 400–408.

Asif Ekbal, Sudip Kumar Naskar, and Sivaji Bandyopadhyay. 2006. A modified joint source-channel model for transliteration. In *Proceedings of the COLING/ACL poster sessions*, pages 191–198, Sydney, Australia. Association for Computational Linguistics.

Swati Gupta. 2004. Aligning Hindi and Urdu bilingual corpora for robust projection. Masters project dissertation, Department of Computer Science, University of Sheffield.

Ulf Hermjakob, Kevin Knight, and Hal Daumé III. 2008. Name translation in statistical machine translation - learning when to transliterate. In *Proceedings of ACL-08: HLT*, pages 389–397, Columbus, Ohio. Association for Computational Linguistics.

Bushra Jawaid and Tafseer Ahmed. 2009. Hindi to Urdu conversion: beyond simple transliteration. In *Conference on Language and Technology 2009*, Lahore, Pakistan.

Mehdi M. Kashani, Eric Joanis, Roland Kuhn, George Foster, and Fred Popowich. 2007. Integration of an Arabic transliteration module into a statistical machine translation system. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 17–24, Prague, Czech Republic. Association for Computational Linguistics.

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

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, Demonstration Program*, Prague, Czech Republic.

Philipp Koehn. 2004a. Pharaoh: A beam search decoder for phrase-based statistical machine translation models. In *AMTA*, pages 115–124.

Philipp Koehn. 2004b. Statistical significance tests for machine translation evaluation. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 388–395, Barcelona, Spain, July. Association for Computational Linguistics.

Haizhou Li, Zhang Min, and Su Jian. 2004. A joint source-channel model for machine transliteration. In *ACL '04: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, pages 159–166, Barcelona, Spain. Association for Computational Linguistics.

M G Abbas Malik, Christian Boitet, and Pushpak Bhattacharyya. 2008. Hindi Urdu machine transliteration using finite-state transducers. In *Proceedings of the 22nd International Conference on Computational Linguistics*, Manchester, UK.

Robert C. Moore. 2002. Fast and accurate sentence alignment of bilingual corpora. In *Conference of the Association for Machine Translation in the Americas (AMTA)*.

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

Kishore A. Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. BLEU: a method for automatic evaluation of machine translation. Technical Report RC22176 (W0109-022), IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY.

Ari Pirkola, Jarmo Toivonen, Heikki Keskustalo, Kari Visala, and Kalervo Järvelin. 2003. Fuzzy translation of cross-lingual spelling variants. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 345–352, New York, NY, USA. ACM.

R. Mahesh K. Sinha. 2009. Developing English-Urdu machine translation via Hindi. In *Third Workshop on Computational Approaches to Arabic Script-based Languages (CAASL3), MT Summit XII*, Ottawa, Canada.

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *Intl. Conf. Spoken Language Processing*, Denver, Colorado.

Paola Virga and Sanjeev Khudanpur. 2003. Transliteration of proper names in cross-lingual information retrieval. In *Proceedings of the ACL 2003 workshop on Multilingual and mixed-language named entity recognition*, pages 57–64, Morristown, NJ, USA. Association for Computational Linguistics.

Bing Zhao, Nguyen Bach, Ian Lane, and Stephan Vogel. 2007. A log-linear block transliteration model based on bi-stream HMMs. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 364–371, Rochester, New York. Association for Computational Linguistics.