Substring-based unsupervised transliteration with phonetic and contextual knowledge

Anoop Kunchukuttan[1], **Pushpak Bhattacharyya**[1], **Mitesh M. Khapra**[2]
[1] Department of Computer Science & Engineering,
Indian Institute of Technology Bombay.
{anoopk,pb}@cse.iitb.ac.in
[2] IBM Research India.
mikhapra@in.ibm.com

Abstract

We propose an unsupervised approach for *substring-based* transliteration which incorporates two new sources of knowledge in the learning process: (i) context by learning substring mappings, as opposed to single character mappings, and (ii) phonetic features which capture cross-lingual character similarity via prior distributions.

Our approach is a *two-stage iterative, boot-strapping solution*, which vastly outperforms Ravi and Knight (2009)'s state-of-the-art unsupervised transliteration method and outperforms a rule-based baseline by up to 50% for top-1 accuracy on multiple language pairs. We show that substring-based models are superior to character-based models, and observe that their top-10 accuracy is comparable to the top-1 accuracy of supervised systems.

Our method only requires a phonemic representation of the words. This is possible for many language-script combinations which have a high grapheme-to-phoneme correspondence *e.g.* scripts of Indian languages derived from the Brahmi script. Hence, Indian languages were the focus of our experiments. For other languages, a grapheme-to-phoneme converter would be required.

1 Introduction

Transliteration is a key building block for multilingual and cross-lingual NLP since it is useful for user-friendly input methods and applications like machine translation and cross-lingual information retrieval. The best performing solutions are supervised, discriminative learning methods which learn transliteration models from parallel transliteration corpora. However, such corpora are available only for some language pairs. It is also expensive and time-consuming to build a parallel corpus.

This limitation can be addressed in three ways: **(i)** train a transliteration model on mined parallel transliterations. The transliterations can be mined from monolingual comparable corpora (Jagarlamudi and Daumé III, 2012) or parallel translation corpora (Sajjad et al., 2012). However, it may not be possible to mine enough transliteration pairs to train a system for most languages (Irvine et al., 2010). **(ii)** transliterate via a bridge language (Khapra et al., 2010) when transliteration corpora involving bridge languages is available. **(iii)** learn transliteration models in an unsupervised setting using only monolingual word lists. *Unsupervised transliteration* can be defined as: Learn a transliteration model ($\mathcal{TX}$) from the source language (F) to the target (E) language given their respective monolingual word lists, W_F and W_E respectively. We explore this direction in the present work, addressing shortcomings in the previous work (Ravi and Knight, 2009; Chinnakotla et al., 2010).

Our work addresses two major limitations in existing unsupervised transliteration approaches: (i) lack of linguistic signals to drive the learning, and (ii) limited use of context since their model is character-based. Due to this knowledge-lite approach, these model performs poorly. Our **primary contributions** are novel methods to incorporate two knowledge sources, phonetic and contextual, in the training process. These knowledge sources are critical since statistical co-occurrence signals used in supervised learning are not available for unsupervised learning. Unlike transliteration mining, our approach can learn effectively even if the source and target corpus do not have any transliteration pairs in common.

We propose a *two-stage* iterative, bootstrapping

Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL), pages 270–279,
Berlin, Germany, August 7-12, 2016. ©2016 Association for Computational Linguistics

approach for learning unsupervised transliteration models. In the first stage, a character-based model is learnt which is used to bootstrap and learn a series of improved substring-based models in the second stage.

The first stage incorporates two linguistic signals to drive the learning process: **phonemic correspondence** and **phonetic similarity**. This means we make the model aware that two characters represent either the same phoneme (क in Hindi and ক in Bengali [IPA: k]) or similar phonemes (क [IPA: k] in Hindi and খ [IPA: k^h] in Bengali - which differ only in aspiration). We achieve this by incorporating phonetic information as **prior distributions** in our EM-MAP approach to character-based unsupervised learning. We show that these linguistic signals can improve top-1 accuracy by 20%-100% over a baseline rule-based system. It is also vastly superior to knowledge-lite unsupervised methods.

The second stage incorporates **contextual knowledge** by unsupervised learning of a substring-based transliteration model *viz.* learning mappings from substring in one language to another, as opposed to learning single character mappings. In other words, along with learning mappings of the form ($k_{hindi} \rightarrow k_{bengali}$), we also try to learn mappings of the form ($kaa_{hindi} \rightarrow kaa_{bengali}$). It is known that substring-based transliteration outperforms character-based transliteration in a supervised setting due to the additional context information (Sherif and Kondrak, 2007). To the best of our knowledge, ours is the **first unsupervised approach for substring-based transliteration**. It outperforms a character-based model by up to 11% in terms of top-1 accuracy and 27% in terms of top-10 accuracy.

The top-10 accuracy of our unsupervised system is comparable to the top-1 accuracy of a supervised system. Hence, the unsupervised system may be a reasonable substitute for supervised systems in applications which require transliteration (*e.g.* handling untranslated words in MT) and can disambiguate from the top-k transliterations with information available to the application (*e.g.* LM in MT systems).

The focus of our work was Indian languages using scripts descended from the ancient Brahmi script. We show that our methods can be applied to these languages **without requiring phoneme dictionaries or grapheme-to-phoneme converters**. We achieve this by using **scriptural properties and similarity across scripts** to capture phonemic correspondence and phonetic similarity, and show results on 4 languages using 4 different scripts. At least 19 of the Indian subcontinent's top 30 and 9 of the top 10 most spoken languages use Brahmi-derived scripts. Each of these languages have more than a million speakers with an aggregate speaker population of about 900 million, so our method is widely applicable.

2 Related Work

Unsupervised transliteration has not been widely explored. Chinnakotla et al. (2010) generate transliteration candidates using manually developed character mapping rules and rerank them with a character language model. The major limitations are: (i) character transliteration probability is not learnt, so there is undue reliance on the language model to handle ambiguity, and (ii) significant manual effort for good coverage of mapping rules.

Ravi and Knight (2009) propose a decipherment framework based approach (Knight et al., 2006) to learn phoneme mappings for transliteration without parallel data. In theory, it should be able to learn transliteration probabilities and is a generalization of Chinnakotla et al. (2010)'s approach. But its performance is very poor due to lack of linguistic knowledge and has a reasonable performance only when a unigram word-level LM is used. This signal essentially reduces the approach to a lookup for the generated transliterations in a target language word list; the method resembles transliteration mining. It will perform well only if the unigram LM has a good coverage of all named entities in the source word list. For morphologically rich target languages, it may be difficult to find the exact surface words in the unigram LM.

Our character level model approach is a further generalization of Ravi and Knight (2009)'s work since it also allows modelling of prior linguistic knowledge in the learning process. This overcomes the most significant gap in their work.

Some approaches to transliteration mining are also relevant to the present work. Tao et al. (2006) show improvement in transliteration mining performance using phonetic feature vectors resembling the ones we have used. Jagarlamudi and Daumé III (2012) use phonemic representa-

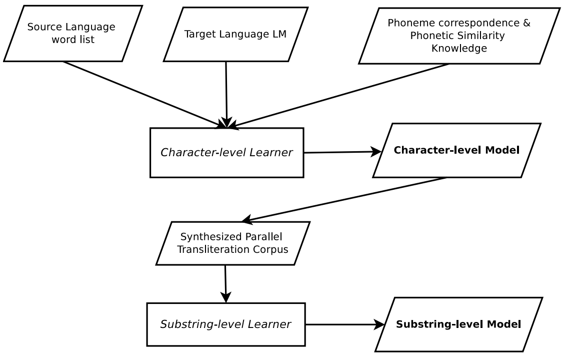
Source Language word list
Target Language LM
Phoneme correspondence & Phonetic Similarity Knowledge
Character-level Learner
Character-level Model
Synthesized Parallel Transliteration Corpus
Substring-level Learner
Substring-level Model

Figure 1: Overview of Proposed Approach

tion based interlingual projection for multilingual transliteration mining. To the best of our knowledge, ours is the first work to use phonetic feature vectors for transliteration as opposed to transliteration mining.

We use a substring-based log-linear model in our second stage. There are some parallels to this approach in the transliteration mining litereature. Some transliteration mining approaches have used a log-linear classifier to incorporate features to distinguish transliterations from non-transliterations (Klementiev and Roth, 2006; Chang et al., 2009). Sajjad et al. (2011) use a substring-based log-linear model trained on a noisy, intermediate transliteration corpus to iteratively remove *bad* (low-scoring) transliteration pairs found in the discovery process.

3 Unsupervised Substring-based Transliteration

In this section, we give a high-level overview of our approach for learning a substring-based transliteration model in an unsupervised setting (depicted in Figure 1). The inputs are monolingual lists of words, W_F and W_E, for the source (F) and target (E) languages respectively. Note that these are neither parallel nor comparable lists. We need a **phonemic representation** of the words.

For Indic scripts, which are used in our experiments, we use the orthographic representation itself as the phonemic representation since there is high grapheme to phoneme correspondence. Hence, we use the terms *character* and *phoneme* interchangeably.

The training is a two-stage process as described below. First, character mappings are learnt followed by learning of substring mappings by bootstrapping the character-based model. This process is analogous to phrase-based statistical machine translation, where phrase pairs are extracted from word aligned sentence pairs.

Stage One: In the first stage, character transliteration probabilities are learnt from the monolingual word lists.

In a supervised setting, an EM algorithm using maximum likelihood estimation (EM-MLE) (Knight and Graehl, 1998) exploits co-occurrence of characters to learn model parameters. But, prior linguistic knowledge has to be incorporated for effective learning in an unsupervised setting. Hence, we propose an unsupervised Expectation Maximization with Maximum Aposteriori estimation framework (EM-MAP) for learning the character-level transliteration model. Linguistic knowledge is incorporated in the form of prior distributions on the model parameters in this framework. The details of the model and choice of prior distributions are described in Section 4.

We considered two linguistic signals for designing the prior distributions. The first is **phonemic correspondence** *i.e.* characters in the two languages representing the same phoneme. *e.g.* The characters क (ka) in Hindi and ক (ka) Bengali represent the same phoneme (IPA: k). But phonemic correspondence cannot account for phonemes which differ only by some phonetic features. *e.g.* vowel length (short इ [i] in Hindi, long ঈ [I] in Bengali), aspiration (unaspirated क in Hindi [IPA: k], aspirated খ in Bengali [IPA: k^h]). Such transformations are common during transliteration, so we use **phonetic similarity** as our second linguistic signal.

Stage Two: In the second stage, we learn a discriminative, log-linear model with **arbitrary substrings as the unit of transliteration**. For learning the substring based model, a **pseudo-parallel** transliteration corpus is first synthesized using the character-level model. We discuss Stage 2 in detail in Section 5.

We illustrate the need for substring level models with an example. In Indic scripts, the *anusvaara* (nasalization diacritic) can map to any of the 5 nasal consonants depending on the consonant following the *ansuvaara* in the source word. So, in the hi-bn pair (चंबल [*ca.mbala*], চম্বল [*cambala*]), the *anusvaara* (.m) maps to the nasal consonant (m) since the next character is the labial consonant *ba*. This shows the need for contextual information to resolve transliteration ambiguities. Substring-

Algorithm 1 Train character-level model

```
procedure UNSUP-CHAR($W_F, LM_E$)
        ▷ $LM_E$: char-level language model for E
    $\Theta \leftarrow$ initialize-params()
    $i \leftarrow 0$
    while $i \leq N$ do ▷ N: Number of iterations
        $W_{E'} \leftarrow$ c-decode($W_F, \Theta, LM_E$)
        $A \leftarrow$gen-alignments($W_F, W_{E'}$)
        $\Theta \leftarrow \arg\max_\Theta \mathcal{Q}'_{W_F}(\Theta)$    ▷ m-step
        if converged($\Theta$) then
            break
        $\sigma \leftarrow$ e-step($A, \Theta$)
    return $\Theta$
```

based models, which learn substring mappings like *.mba* → *mba*, are one way to incorporate contextual information and have been shown to perform better in a supervised setting (Sherif and Kondrak, 2007). Contextual information is especially important in an unsupervised setting.

4 Character-based Unsupervised Transliteration

In the first stage, we **learn character transliteration probabilities** from monolingual word lists. The generative story for the training data (the source language corpus, W_F) is explained below using a noisy channel model.

An unknown target language word **e** is selected using a language model $P(\mathbf{e})$. The target word is transformed to a source language word **f** by a channel whose properties are represented by the transliteration probability distribution (Θ). *The target language word* **e** *is a latent variable in the unsupervised setting*, so we need to compute the expectation over all possible values of **e**.

$$P(\mathbf{f}) = \sum_{\mathbf{e}} P(\mathbf{f}|\mathbf{e})P(\mathbf{e}) \qquad (1)$$

We use Knight and Graehl (1998)'s transliteration model where, the word pair $(\mathbf{f}, \mathbf{e})$ is generated by successively selecting one or more source characters f for each target character e as per a latent alignment **a** with probability $P(f|e) = \theta_{f,e}$. We restrict our model to 1-1 and 1-2 character mappings from target to source characters. The **likelihood of a single training instance** is given by:

$$L(\Theta) = \sum_{\mathbf{e}} P(\mathbf{e}) \sum_{\mathbf{a}} \prod_{i=1}^{|\mathbf{e}|} \theta_{f_{a_i},e_i} \qquad (2)$$

In the supervised framework of Knight and Graehl (1998), parameters are learnt using the EM algorithm where the alignment structure **a** is the latent variable. The discovery of hidden alignments helps compute the transliteration probabilities based on co-occurrence of characters. In the absence of parallel corpora, co-occurrence is no longer a learning signal and it is not possible to learn the character transliteration probabilities reliably. To compensate for this, we define **Dirichlet priors** (D_e) over each character transliteration probability distributions (Θ_e), which can be used to encode linguistic knowledge. This leads to our proposed **EM-MAP training objective** for the M-step over the entire training set (W_F).

$$\mathcal{Q}_{W_F}(\Theta) = \sum_{\mathbf{f} \in W_F} \Bigg\{ \sum_{\mathbf{e}} \Big\{ \delta_{\mathbf{e},\mathbf{f}} \sum_{\mathbf{a}} \Big[\sigma_{\mathbf{a},\mathbf{f},\mathbf{e}} \sum_{f,e} n_{f,e,\mathbf{a}} \log \theta_{f,e} \Big] + \log P(\mathbf{e}) \Big\} \Bigg\} + \sum_{e \in C_E} \log D_e(\alpha_{f_1,e} ... \alpha_{f_{|C_F|},e}) \qquad (3)$$

s.t

$$\forall e \in C_E, \sum_{j=1}^{j=|C_F|} \theta_{f_j,e} = 1$$

where,
$\delta_{\mathbf{e},\mathbf{f}} = P(\mathbf{e}|\mathbf{f})$, $\sigma_{\mathbf{a},\mathbf{f},\mathbf{e}} = P(\mathbf{a}|\mathbf{e}, \mathbf{f})$ are conditional probabilities of the latent variables computed in the E-step. These are computed using the previous iteration's parameter values, whose values are fixed in the current iteration.
$n_{f,e,\mathbf{a}}$ is the number of times characters e and f are aligned in the alignment structure **a**.
C_F and C_E are the character sets of the source and target languages respectively.

In the unsupervised setting, the target word (**e**) is also an latent variable. As seen in Equation 3, the M-step requires computing an expectation over all latent variables (target word and alignments). Given the target word, it is possible to enumerate all alignments of the word pairs, but it is not possible to enumerate all possible strings (**e**). *Hence, we approximate the expectation over* **e** *by a* ***max*** *operation* (the so-called Viterbi approximation).

The modified objective ($\mathcal{Q}'_{W_F}$) for the M-step effectively means: With the current set of parameter values, we decode the source words to generate the target words, creating a synthetic parallel transliteration corpus. Then, the M-step updates can be done using Knight and Graehl (1998)'s supervised framework. The resulting update equation for the transliteration probabilities in the **M-step** is:

$$\theta_{f,e} = \frac{1}{\lambda_e}\left\{\alpha_{f,e} - 1 + \sum_{\mathbf{f}\in W_F}\sum_{\mathbf{a}} \sigma_{\mathbf{a},\mathbf{f},\mathbf{e}}\, n_{f,e,\mathbf{a}}\right\} \quad (4)$$

where, λ_e is a normalizing factor and **e** is the best transliteration of **f** as per the previous iteration's parameters. Note that $\delta_{\mathbf{e},\mathbf{f}}$ plays no role in $(\mathcal{Q}'_{W_F})$.

The **E-step** update to compute the conditional probabilities of the latent alignment variables is given by:

$$\sigma_{\mathbf{a},\mathbf{f},\mathbf{e}} = \frac{1}{Z} \times \prod_{i=1}^{|\mathbf{e}|} \theta_{f_{a_i},e_i} \quad (5)$$

where, Z is a normalizing factor.

The training procedure can thus be understood to follow a *decode-train-iterate* paradigm. Algorithm 1 shows the procedure for character level training. In each iteration, a pseudo-parallel corpus by decoding W_F using the current set of parameters (Line 5, Viterbi approximation) from which updated parameters are learnt (Line 7) and alignment probabilities recomputed (Line 10).

4.1 Linguistically Informed Priors:

In this section, the different prior distributions we designed to encode phonetic knowledge about characters is described. These are instantiations of Dirichlet priors (D_e), which serve as a conjugate prior to the multinomial distribution ($\theta_{f,e}$). The hyperparameters ($\alpha_{f,e}$) of D_e determine the nature of the prior distribution. They can be interpreted as additional, virtual alignment counts of the character pair (f, e) for maximum likelihood estimation of $\theta_{f,e}$.

Phoneme Correspondence (PC) Prior: This simply establishes a one-one correspondence (denoted by $\hat{=}$) between the same phonemes (or characters representing the same phonemes). It does not capture the notion of similarity between characters.

$$\alpha_{f,e} = \beta \quad \text{if} f \mathrel{\hat{=}} e \quad (6)$$

$$= 0.01 \quad \text{elsewhere} \quad (7)$$

Phonemic correspondence is also used to initialize the transliteration probabilities for the EM algorithm (*PC Init*):

$$\theta^{\text{init}}_{f,e} = \frac{\alpha_{f,e}}{\sum\limits_{x\in C_F}\alpha_{x,e}} \quad (8)$$

Basic Character Type:	vowel, consonant, anusvaara, nukta, halanta, others
Vowel Features	
Length:	short, long
Strength:	weak, medium, strong
Status:	Independent, Dependent
Horizontal position:	Front, Back
Vertical position:	Close, Close-Mid, Open-Mid, Open
Lip roundedness:	Close, Open
Consonant Features	
Place of Articulation:	velar, palatal, retroflex, dental, labial
Manner of Articulation:	plosive, fricative, flap, approximant (central or lateral)
Aspiration, Voicing, Nasal:	True, False

Table 1: Phonetic features for Indic scripts

Phonetic Similarity Priors This prior captures similarity between two phonemes based on their phonetic properties. The phonetic properties of a phoneme can be encoded as a bit vector (v) as explained in Section 4.2. We experimented with two priors based on phonetic properties.

- Cosine Prior: It is based on the cosine similarity between the two bit vectors.

$$\alpha_{f,e} = \gamma_c \cos(v_f, v_e) \quad (9)$$

- Sim1 Prior: Cosine similarity tends to produce very diffused transliteration probability distributions. We propose a modified prior (called *sim1*) which tries to alleviate this problem by making the phonemic differences sharper.

$$\alpha_{f,e} = \gamma_s \frac{5^{v_f \cdot v_e}}{\sum\limits_{x\in C_F} 5^{v_x \cdot v_e}} \quad (10)$$

β, γ_c and γ_s are scale factors for the Dirichlet distribution.

4.2 Extracting Phonetic Features for Indic scripts

We now describe a method for deriving phonetic correspondences and constructing phonetic features vectors for Indic scripts. Indic scripts generally have a one-one correspondence from characters to phonemes in the scripts. Hence, each character is represented by a feature vector representing its phonetic properties as described in Table

1. The feature vector is represented as a bit vector with a bit for each value of every property.

The logical character set is roughly the same across all Indic scripts, though the visual glyph varies to a great extent. So phonemic correspondence can be easily determined for Unicode text since the first 85 characters of all Indic scripts are aligned to each other by virtue of having the same offset from the start of the script's codepage. These cover all commonly used characters. There are a few exceptions to this simple mapping scheme, most of which can be handled using simple rules.

Notable among these is the Tamil script, which does not have characters for aspirated as well as voiced plosives, so the corresponding unvoiced, unaspirated plosive characters are used to represent these sounds too. In the phonetic feature representation of such characters for Tamil, both the voiced as well as unvoiced bits and aspirated/unaspirated bits are set on, reflecting the ambiguity in the grapheme-to-phoneme mapping.

5 Bootstrapping substring-based models

In the second stage, we train a **discriminative, log-linear transliteration model which learns substring mappings**. We use the log-linear model proposed by Och and Ney (2002) for statistical machine translation and analogous transliteration features. The features are: substring transliteration probabilities, weighted average character transliteration probabilities and character language model score. The conditional probability of the target word **e** given the source word **f** is:

$$P(\mathbf{e}|\mathbf{f}) = \prod_{i=1}^{NP} P(\bar{e_i}|\bar{f_i}) = \prod_{i=1}^{NP} \exp \sum_{j=0}^{NF} \lambda_j g_j(\bar{f_i}, \bar{e_i}) \tag{11}$$

where, $\bar{f_i}$ and $\bar{e_i}$ are source and target substrings respectively, λ_j and g_j are feature weight and feature function respectively for feature j, NP number of substrings and NF is number of features.

We synthesize a **pseudo-parallel transliteration corpus** (W_F, $W_{E'}$) for training the discriminative model by transliterating the source language words (W_F) using the character level model from the first stage. Since the top-1 transliteration may be incorrect, we consider the top-k transliterations to improve the odds that the pseudo-parallel corpus contains the correct transliteration. For **training**, the pseudo-parallel corpus contains k transliteration pairs for every source language word. For **tuning** the feature weights, we use a small held-out set of top-1 transliteration pairs from the pseudo-parallel corpus, since it likely to be the most accurate one.

We run **multiple iterations** of the discriminative training process, with each being trained on the pseudo-parallel corpus synthesized using the previous iteration's models. The models in subsequent iterations are **bootstrapped** from the earlier models. The training continues for a fixed number of iterations although other convergence methods can also be explored. Like the MAP-EM solution for the first stage, the second stage also uses a *decode - train - iterate* paradigm for learning a substring-based model.

6 Experiments

Data: We experimented on the following Indian language pairs representing two language families: Bengali→Hindi, Kannada→Hindi, Hindi→Kannada and Tamil→Kannada. Bengali (bn) and Hindi (hi) are Indo-Aryan languages, while Kannada (kn) and Tamil (ta) are Dravidian languages. We used 10k source language names as training corpus, which were collected from various sources.

We evaluated our systems on the NEWS 2015 Indic dataset. We created this set from the English to Indian language training corpora of the NEWS 2015 shared task (Banchs et al., 2015) by mining name pairs which have English names in common. 1500 words were selected at random to create the testset. The remaining pairs are used to train and tune a skyline supervised transliteration system for comparison. The training sets are small, the number of name pairs being: 2556 (bn-hi), 4022 (kn-hi), 3586 (hi-kn) and 3230 (ta-kn).

Experimental Setup: We trained the character level unsupervised transliteration systems with source language word lists using a custom implementation [1]. We set the the value of the scaling factors ($\beta, \gamma_c, \gamma_s$) to 100. Viterbi decoding was done with a bigram character language model, followed by re-ranking with a 5-gram character language model.

We trained the substring level discriminative transliteration models as well as a skyline su-

[1] `https://github.com/anoopkunchukuttan/transliterator`

Method	bn-hi			hi-kn			kn-hi			ta-kn		
	A_1	F_1	A_{10}	A_1	F_1	A_{10}	A_1	F_1	A_{10}	A_1	F_1	A_{10}
PC_Init	12.72	68.95	18.94	0.00	44.76	0.07	0.20	48.84	0.54	0.00	44.46	0.27
Rule	16.13	74.60	16.13	**13.75**	**79.67**	13.75	12.90	79.29	12.90	10.25	68.49	10.25
Initialization: PC_Init+												
PC_Prior	**18.27**	**75.50**	27.04	12.53	77.32	17.89	**27.69**	**81.06**	**43.55**	13.49	**69.85**	**29.06**
Cosine Prior	17.74	75.09	26.57	11.38	75.08	18.09	17.54	77.69	32.86	13.21	69.44	26.64
Sim1 Prior	18.07	75.25	**29.05**	11.72	75.61	**20.26**	19.69	78.18	37.84	**13.55**	69.74	28.19
Supervised	32.06	83.03	63.32	30.01	85.93	69.37	54.23	90.05	80.04	30.74	81.62	64.33

Table 2: Results for character-based model (% scores)

pervised transliteration system using the *Moses* (Koehn et al., 2007) machine translation system with default parameters. Batch MIRA (Cherry and Foster, 2012) was used to tune the Stage 2 systems with 1000 name pairs and supervised systems with 500 name pairs. The tuning set for the Stage 2 systems were drawn from the the top-1 transliterations in the synthesized, pseudo-parallel corpus; no true parallel corpus is used. Monotone decoding was performed. We used a 5-gram character language model trained with Witten-Bell smoothing on 40k names for all target languages. We ran Stage 2 for 5 iterations.

For a rule-based baseline, we used the script conversion method implemented in the *Indic NLP Library*[2] (Kunchukuttan et al., 2015) which is based on phonemic correspondences.

Evaluation: We used top-1 accuracy based on exact match (A_1) and Mean F-score (F_1) at the character level as defined in the NEWS shared tasks as our evaluation metrics (Banchs et al., 2015). We also used top-10 accuracy as an evaluation metric (A_{10}), since applications like MT and IR can further disambiguate with context information available to these applications.

7 Results and Discussion

Table 2 shows the results for the rule-based system and various character-based unsupervised models. Table 3 shows results for substring-level models bootstrapped from different character-based models. Results of supervised transliteration on a small training set are also shown in both tables.

Baseline models: Parameter initialization with phoneme correspondence mappings and add-one smoothing prior (*PC_Init*) is comparable to Ravi and Knight (2009)'s method and performs very poorly as reported in their work too. We also experimented with re-ranking the results using a unigram word based LM - our approximation to Ravi and Knight (2009)'s use of a word based LM - and its accuracy is comparable to *PC_Init*. The unigram LM was trained on a corpus of 185 million and 42 million tokens for *hi* and *kn* respectively. Thus, this knowledge-lite approach cannot learn a transliteration model effectively.

Rule-based transliteration (*Rule*) performs significantly better than *PC_Init*. The phonetic nature of Indic scripts makes the rule-based system a stronger baseline, yet this simple approach does not ensure high accuracy transliteration. Phonetic changes like changes in manner/place of articulation, voicing, *etc.* make transliteration non-trivial and phonetic correspondence is not sufficient to ensure good transliteration.

Effect of linguistic priors: The addition of linguistically motivated priors (*PC_Prior, Cosine_Prior, Sim1_Prior*) significantly improves the transliteration accuracy over the *PC_Init* approach. There is a significant improvement in top-1 accuracy over the *Rule* approach too for 2 language pairs (12%, 31% and 133% increase for bn-hi, ta-kn and kn-hi pairs respectively), but a drop of 9% for the hi-kn pair. A major reason for lower accuracy of hi→kn pair is an important difference between Kannada and Hindi writing conventions. Unlike Hindi, Kannada assumes an implicit vowel 'A' at the end of a word unless another vowel or nasal character terminates the word. Therefore, the vowel suppressor character (*halanta*) must be generated during hi→kn transliteration. Our method is poor at this generation, but conversely it does better at deletion of *halanta* for kn→hi transliteration. There is substantial improvement for the ta-kn pair also, even though there are some grapheme-to-phoneme ambiguities in the Tamil script.

In general, the phonemic correspondence prior results in better top-1 accuracy, whereas priors us-

[2] `http://anoopkunchukuttan.github.io/indic_nlp_library`

Stage1 Model	Iterations	bn-hi		hi-kn		kn-hi		ta-kn	
		A_1	A_{10}	A_1	A_{10}	A_1	A_{10}	A_1	A_{10}
PC_Init	1	14.12	22.89	0.00	0.06	0.53	4.50	0.07	1.35
	5	15.26	25.43	0.00	0.47	1.01	8.53	0.40	2.76
PC_Init+PC_Prior	1	18.74	29.38	13.21	19.31	**28.43**	44.96	15.31	32.23
	5	19.34	32.73	13.21	20.39	21.10	45.03	**19.29**	37.15
PC_Init+Cosine prior	1	18.86	29.65	12.40	20.94	17.94	39.92	15.71	32.91
	5	18.94	31.33	12.94	23.92	16.73	44.76	18.88	36.08
PC_Init+Sim1 prior	1	19.28	34.34	12.6	23.85	19.62	47.98	16.99	34.86
	5	**20.54**	**37.61**	**13.82**	**25.88**	18.55	**50.27**	18.95	**38.50**
Supervised		32.06	63.32	30.01	69.37	54.23	80.04	30.74	64.33
Mined Pairs		26.97	51.34	-	-	-	-	-	-
Bridge Languages		25.97	58.23	18.22	52.85	33.60	67.88	13.01	42.28

Table 3: Results for substring-based model (% scores)

ing phonetic similarity give better top-10 accuracy. The phonetic similarity based priors are smoother compared to the sparse *PC_Prior* since they capture character similarity. This allows them to discover more character mappings, resulting in better top-10 accuracy at the cost of a drop in top-1 accuracy. The sparse *sim1* prior outperforms the smoother cosine similarity prior.

Effect of learning substring mappings: Substring-based transliteration improves the top-1 as well as top-10 accuracy significantly over the underlying character-based models. Across languages, the best substring-based models improve top-1 accuracy by upto 11% and top-10 accuracy by upto 25% over the best character-based models. Therefore, it is clear that contextual information can be harnessed in an unsupervised setting to substantially improve transliteration accuracy.

The iterative procedure is beneficial and transliteration accuracy increases as improved models are built in successive iterations. Large gains are particularly observed in top-10 accuracy.

While *PC_Prior* gives best top-1 accuracy results at the character-level, substring-based models bootstrapped from the *Sim1* prior give better results for both top-1 and top-10 accuracy metrics. Since the *Sim1* prior based character model has better top-10 accuracy, the pseudo-parallel corpus created using this mode is likely to be better than one created using *PC_Prior*. We also observe that substring-based models built without using phonetic priors cannot improve over much over the baseline transliteration.

Illustrative Examples:

- Phonetic similarity based priors were able to discover mappings between similar phonemes. *e.g.* The Bengali word ওয়েস্ট (keolAdeo) was correctly transliterated to the Hindi word केवलादेव (kevalAdeva), due to discovery of similarity between the labial sounds (v) and (o).
- Substring-based models made use of the context to learn the correct transliteration. *e.g.*The Bengali word কেওলাদেও (oyaesTa) was correctly transliterated to the Hindi word वेस्ट (vesta), since the model learnt the substring mapping *oya*→*ve*.
- Substring-based models could make the right choice between short and long vowels (a major source of errors in character-based models).

Comparison with supervised system and some resource-constrained approaches: We compared our best substring-based model (based on *sim1* prior) with a supervised system and the following resource-constrained transliteration systems built using: **(i) Mined pairs** from a translation corpus: We experimented with *bn-hi* on the *Brahminet* mined pairs corpus (Kunchukuttan et al., 2015). Mined corpora involving *kn* were not available. **(ii) Bridge languages:** We used the NEWS 2015 corpus for our experiments with English as the bridge language. (Results in Table 3).

The accuracies of this substring-based system are less than the accuracies of the other methods. This is not unexpected since these methods use more parallel resources than the substring-based approach. But, note that the top-10 accuracy of the substring-based model is comparable to the top-1 accuracies of other approaches. Hence, the substring-based model may be sufficient for an application, like MT or cross-lingual IR, which uses the output of a transliteration system. In MT, untranslated words are replaced by their top-k transliterations. A language model can choose among the transliterations based on context while re-ranking resulting candidate sentences.

Applicability to different languages and scripts: Our experiments span 4 widely-spoken languages from the two major language families in the Indian subcontinent (Indo-Aryan [IA] and Dravidian [DR]). All languages use different Brahmi-descended scripts. We show improvements in IA→IA, DR→DR, DR→IA and IA→DR transliteration.

The first stage leverages the phonetic nature of Indic scripts to obtain phonetic representations from orthographic representations. There are at least 19 languages in India where this condition holds, so at least 171 language pairs can use this approach without grapheme to phoneme converters. Many other script/language pairs also show high grapheme-phoneme correspondence[3]. Our method, though, can also be applied to non-phonetic scripts also by representing the training data at the phonemic level using grapheme-to-phoneme converters.

The second stage makes no assumptions about language or script.

8 Conclusion and Future work

We show that *unsupervised transliteration can substantially benefit from contextual and rich phonetic knowledge*. Phonetic knowledge is incorporated through a novel design of prior distributions for character-level learning, while context is incorporated via substring-based learning. The top-10 accuracy of our systems is comparable with top-1 accuracy of supervised systems, but *requires only monolingual resources*: word lists for Indic languages using Brahmi-derived scripts or phoneme dictionaries for other languages.

In future, we plan to evaluate our method for non-Indic languages and for languages that are less related than the ones studied in this work. Possibilities for improvement include incorporation of phonetic knowledge while learning substring mappings (Stage 2) and better handling of noisy transliterations in the bootstrapping process. It would also be interesting to compare our method with other unsupervised log-linear learning methods like contrastive estimation (Smith and Eisner, 2005).

We would like to note the potential for harnessing similarities among languages for statistical NLP, especially in an unsupervised setting, as demonstrated by our use of similarity among Indic scripts. Finally, the iterative, bootstrapping framework may be useful for unsupervised translation.

[3] `http://www.omniglot.com` lists grapheme-phoneme correspondences for many language/script combinations

References

Rafael E. Banchs, Min Zhang, Xiangyu Duan, Haizhou Li, and A Kumaran. 2015. Report of news 2015 machine transliteration shared task. In *The ACL/IJCNLP 2015's Named Entity Workshop*.

Ming-Wei Chang, Dan Goldwasser, Dan Roth, and Yuancheng Tu. 2009. Unsupervised constraint driven learning for transliteration discovery. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Colin Cherry and George Foster. 2012. Batch tuning strategies for statistical machine translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Manoj K Chinnakotla, Om P Damani, and Avijit Satoskar. 2010. Transliteration for resource-scarce languages. *ACM Transactions on Asian Language Information Processing (TALIP)*.

Ann Irvine, Chris Callison-Burch, and Alexandre Klementiev. 2010. Transliterating from all languages. In *Proceedings of the Conference of the Association for Machine Translation in the Americas (AMTA)*.

Jagadeesh Jagarlamudi and Hal Daumé III. 2012. Regularized interlingual projections: evaluation on multilingual transliteration. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics.

Mitesh M. Khapra, A. Kumaran, and Pushpak Bhattacharyya. 2010. Everybody loves a rich cousin: An empirical study of transliteration through bridge languages. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Alexandre Klementiev and Dan Roth. 2006. Weakly supervised named entity transliteration and discovery from multilingual comparable corpora. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*.

Kevin Knight and Jonathan Graehl. 1998. Machine transliteration. *Computational Linguistics*.

Kevin Knight, Anish Nair, Nishit Rathod, and Kenji Yamada. 2006. Unsupervised analysis for decipherment problems. In *Proceedings of the COLING/ACL* *on Main conference poster sessions*. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*.

Anoop Kunchukuttan, Ratish Puduppully, and Pushpak Bhattacharyya. 2015. Brahmi-net: A transliteration and script conversion system for languages of the indian subcontinent.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*.

Sujith Ravi and Kevin Knight. 2009. Learning phoneme mappings for transliteration without parallel data. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Hassan Sajjad, Alexander Fraser, and Helmut Schmid. 2011. An algorithm for unsupervised transliteration mining with an application to word alignment. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*.

Hassan Sajjad, Alexander Fraser, and Helmut Schmid. 2012. A statistical model for unsupervised and semi-supervised transliteration mining. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*.

Tarek Sherif and Grzegorz Kondrak. 2007. Substring-based transliteration. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*.

Noah A Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*.

Tao Tao, Su-Youn Yoon, Andrew Fister, Richard Sproat, and ChengXiang Zhai. 2006. Unsupervised named entity transliteration using temporal and phonetic correlation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*.