Transliteration for Resource-Scarce Languages

MANOJ K. CHINNAKOTLA, OM P. DAMANI, and AVIJIT SATOSKAR
Indian Institute of Technology Bombay

Today, parallel corpus-based systems dominate the transliteration landscape. But the resource-scarce languages do not enjoy the luxury of large parallel transliteration corpus. For these languages, rule-based transliteration is the only viable option. In this article, we show that by properly harnessing the monolingual resources in conjunction with manually created rule base, one can achieve reasonable transliteration performance. We achieve this performance by exploiting the power of Character Sequence Modeling (CSM), which requires only monolingual resources. We present the results of our rule-based system for Hindi to English, English to Hindi, and Persian to English transliteration tasks. We also perform extrinsic evaluation of transliteration systems in the context of Cross Lingual Information Retrieval. Another important contribution of our work is to explain the widely varying accuracy numbers reported in transliteration literature, in terms of the entropy of the language pairs and the datasets involved.

Categories and Subject Descriptors: I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*Language models*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; I.5.1 [**Pattern Recognition**]: Models—*Statistical*

General Terms: Algorithms, Languages, Experimentation, Performance

Additional Key Words and Phrases: Transliteration, character sequence modeling, resource-scarce languages, prefix-based partial match (PPM), cross entropy

ACM Reference Format:
Chinnakotla, M. K., Damani, O. P., and Satoskar, A. 2010. Transliteration for resource-scarce languages. ACM Trans. Asian Lang. Inform. Process. 9, 4, Article 14 (December 2010), 30 pages. DOI = 10.1145/1838751.1838753. http://doi.acm.org/10.1145/1838751.1838753.

1. INTRODUCTION

Transliteration is the process of mapping a written word from a language-script pair to another language-script pair. For example, Hindi words परम and शेयर correspond to English words *param* and *share*, respectively. Transliterating

M. K. Chinnakotla is supported by a research fellowship award from Infosys Technologies Limited. The Tata Research Development and Design Center (TRDDC) supported us with generous grants.
Authors' address: M. K. Chinnakotla, O. P. Damani, and A. Satoskar, Department of Computer Science and Engineering, IIT Bombay, Powai, Mumbai, India - 400076; email: {manoj,damani,avijit199}@cse.iitb.ac.in.

a word from the language of its origin to a foreign language is called *Forward Transliteration*, while transliterating a loan-word written in a foreign language back to the language of its origin is called *Backward Transliteration*. In this article, we focus on general purpose transliteration involving resource-scarce languages for which very large parallel corpora of transliteration pairs do not exist. By general purpose, we mean that the system should do both forward and backward transliteration.

A natural approach to transliteration is to create a set of character mappings, or character sequence mapping rules, between the languages involved. However, transliteration is one of those problems where the accuracy of an answer is not defined solely by algorithmic rules but also by human convention, which can be inconsistent and somewhat ad hoc at times. Therefore, it is only natural that parallel corpus based systems dominate the transliteration research landscape [AbdulJaleel and Larkey 2003; Ekbal et al. 2006; Ganesh et al. 2008; Haizhou et al. 2004; Kumaran and Kellner 2007; Sherif and Kondrak 2007]. Statistical techniques based on large parallel transliteration corpus work well for the resource rich languages but the resource scarce languages do not have the luxury of such resources. For such languages, rule-based transliteration is the only viable option.

In this article, we show that by using just the monolingual resources in conjunction with manually created rule base, one can achieve reasonable transliteration performance when compared to that of baseline statistical systems trained using parallel corpora. We achieve this performance by properly harnessing the power of *Character Sequence Modeling (CSM)*, typically called the Language Model. Our system uses CSM on the source side for word origin identification, a manually generated non-probabilistic character mapping rule base for generating transliteration candidates, and then again uses the CSM on the target side for ranking the generated candidates. We perform a step-by-step fine-tuning of various CSM parameters.

We present the results for general purpose Hindi to English, and English to Hindi transliteration systems. We also perform extrinsic evaluation of transliteration systems in the context of Cross Lingual Information Retrieval. Another important contribution of our work is to explain the widely varying accuracy numbers reported in transliteration literature, in terms of the entropy of the language pairs and the datasets involved.

To further demonstrate the validity of our approach, we have also experimented with a non-Indian language. Our only criterion for selecting the language was that there should be a publicly available character mapping, and a parallel transliteration corpus for evaluation. We found that such resources are available for Persian to English transliteration in Karimi [2008]. Hence, we decided to work with Persian, despite the fact that none of the authors have any knowledge of Persian whatsoever (we cannot even read the Persian script). We also discovered that short vowels are typically absent from the written Persian words [Karimi 2008], thus making the problem even more challenging.

To summarize, we make the following contributions in this article.

—We show how to build a reasonable transliteration system for resource scarce languages that lack large parallel corpora required to train statistical systems. Our system employs only monolingual resources and simple non-probabilistic character mapping.

—Besides intrinsic evaluation, we also perform extrinsic evaluation of transliteration systems in the context of Cross Lingual Information Retrieval (CLIR). To our knowledge such a wide-ranging evaluation on multiple language pairs and data-sets has not been performed earlier for transliteration.

—We provide an entropy based explanation for widely varying transliteration accuracy numbers reported in the literature.

2. RELATED WORK

Existing transliteration generation approaches can be classified along two dimensions: the level at which the bilingual mapping process is modeled and the approach taken to learn the mapping rules. Table I shows this classification.

In Grapheme-based approaches, transliteration is viewed as a process of mapping a grapheme/character sequence from a source language to a target language ignoring the phoneme-level processes involved. In contrast, in phoneme-based approaches, the whole process of mapping from source language graphemes to source language phonemes to target language phonemes to target language graphemes, is modeled. In hybrid approaches, the models learnt using the above two approaches are combined by either interpolating them or by conditioning the grapheme models on corresponding phonemes [Oh et al. 2006].

Both grapheme- and phoneme-based approaches need some mapping rules between corresponding entities. These rules can either be manually created or can be learned by employing statistical techniques. When these rules are created manually, we call them rule-based approaches. As discussed before and shown in Table I, most of the existing systems learn these rules by employing machine-learning techniques. In general these rules are learned from parallel transliteration corpora. In contrast, given our focus on resource-scarce languages, we use manually created bilingual mapping rules and rely on statistical knowledge that can be gleaned from monolingual corpora only. The work described in Ekbal et al. [2006] also combines rule-based and statistical techniques by learning character mappings from parallel corpora and falling back to manually created rule bases to resolve ambiguity. In contrast, we do not use parallel corpora at all and use statistical techniques only for learning monolingual language models.

Different from transliteration generation systems are transliteration mining systems that try to obtain parallel transliteration pairs from comparable corpora [Collier et al. 1997; Klementiev and Roth 2006; Paşca 2007; Sproat

Table I. A Classification of Transliteration Generation Approaches

Modeling Levels	**Learning Approach**		
	Rule Based	**Statistical**	**Combined**
Grapheme	Darwish et al. [2001] (English->Arabic) Malik et al. [2006] (Shahmukh->Gurmukhi) Saini et al. [2008] (Shahmukhi->Gurmukhi) Collier et al. [1997] (English->Japanese)	Ganesh et al. [2008] (Hindi->English), AbdulJaleel and Larkey [2003] (English->Arabic), Sherif and Kondrak [2007] (Arabic->English), Zelenko and Aone [2006] ({Arabic, Korean, Russian}-> English), Huang [2005] (Chinese->English), Haizhou et al. [2004] (English->Chinese), Kumaran and Kellner [2007] (English< − >{Hindi, Tamil, Japanese, Arabic}), Min et al. [2004] (English->Chinese)	AsifEkbal et al. [2006] (Bengali->English) **Current Work**
Phoneme	Arbabi et al. [1994] (Arabic->English), Wan and Verspoor [1998] (English->Chinese), Surana and Singh [2008] (English->Hindi)	Knight and Graehl [1997] (Japanese->English), Virga and Khudanpur [2003] (English->Chinese), Gao et al. [2004] (English->Chinese)	Oh and Choi [2002] (English-> Korean), Kawtrakul et al. [1998] (Thai->English)
Hybrid (Grapheme + Phoneme)		Oh and Choi [2005] & Oh et al. [2006] (English-> {Korean, Japanese}) Yaser Al-Onaizan and Kevin Knight [2002] (Arabic->English), Bilac and Tanaka [2004] (Japanese->English, Chinese-> English)	

et al. 2006; Udupa et al. 2009]. In Klementiev and Roth [2006], transliteration pairs are obtained by doing frequency analysis of words in a weakly temporally aligned comparable corpora. It is assumed that the aligned named entities will have comparable frequency vs. time signatures. In Collier et al. [1997] and Udupa et al. [2009], documents from comparable corpora are first aligned with each other and then named-entity pairs are mined from each pair of document by using a cross language transliteration similarity model. In Sproat et al. [2006], both frequency and similarity models are combined. The transliteration mining approach is supplementary to the transliteration generation approach in that one may always need to transliterate words which have not been mined. Also, while document similarity based methods are shown to perform much better than the frequency based methods [Udupa et al. 2009], unlike the frequency based methods, document similarity based mining method do need to employ some kind of transliteration similarity/generation model to identify potential pairs.

Having contextualized our work, we next describe the evaluation metrics used for measuring system performance.

3. EVALUATION METRICS AND BASELINE APPROACHES

Given a source language word, a transliteration system produces a ranked list of possible candidate answers. Given a set of correct transliterations for that word, we define the answer generated by the transliteration system to be correct at rank k, if one of the top k candidates generated by the system belongs to the set of correct transliterations for that word. The *accuracy* at rank k is defined as the fraction of the test words for which the system generates the correct answer at rank k. Given N input words to be transliterated and list of top k candidates generated for each input word, we use another metric called the Mean Reciprocal Rank (MRR) at rank k defined as follows:

$$MRR(k) = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{Rank(n)}$$

$Rank(n)$ is the rank of the first correct answer for the n^{th} input word in the corresponding list of k candidates. In case, no correct answer is found for a word in the list, its rank is taken to be infinite. We report the Accuracy and MRR figures at Rank 5 throughout the article, unless otherwise stated.

3.1 Evaluation Data

We randomly partition a parallel corpus of 30,000 word pairs (referred to as *30K Dataset*) into training, development, and the test set, the details of which are given in the Table II. To make sure that our partitioning is not biased, we perform a five-fold cross validation on this dataset and all numbers reported on this dataset are the average values obtained from five different partitions. In addition to this, we also use the standard NEWS 2009 Transliteration Shared Task dataset [Li et al. 2009]. The NEWS 2009 dataset contains many multi-word named-entities which were converted into single word source-target transliteration pairs. Note that the word origin information is not used and discussed until Section 6, but we include it in Table II for presenting all related information in a single place.

3.2 Baseline Approaches for Comparison

The following language independent statistical approaches have recently been shown to perform quite well for the transliteration task. We chose them as baseline since they can be implemented easily using off-the-shelf components like CRF++ [Kudo 2003], GIZA++ [Och and Ney 2003], and Moses [Hoang et al. 2007]. Both of the following systems are trained using the training set mentioned in Table II.

3.2.1 *CRF Based Transliteration.* In Ganesh et al. [2008], the transliteration problem is posed as a sequence learning problem and a Conditional Random Field (CRF) based approach is used. They report better accuracies than

Table II. Details of 30,000 and NEWS 2009 Datasets Used for Evaluation of Hindi to English Transliteration

5-fold cross validation was performed on the 30,000 dataset and the word origin statistics given previously are from a single partition. In NEWS 2009 dataset, the standard training, development, and test datasets provided as part of shared task evaluation were used.

	Indo-Arabic Org	**Non Indo-Arabic Org**	**Total**
30K Dataset			
Train	11988	9012	21000
Dev	1705	1295	3000
Test	2811	3189	6000
Total	16504	13496	30000
NEWS 2009			
Train	6327	4832	11159
Dev	840	506	1346
Test	748	254	1002
Total	7915	5592	13507

the HMM based approach [AbdulJaleel and Larkey 2003]. We implemented a modified version of their approach using CRF++. We use two way alignment in GIZA++ for learning the character mappings whereas the original approach uses only one way alignment.

3.2.2 *SMT Based Transliteration.* In Sherif and Kondrak [2007] and Huang [2005] the transliteration problem is posed as a Phrase-Based Machine Translation problem where the words are replaced by the characters. We implemented the above approach using GIZA++ aligner, the Moses decoder, and the SRILM toolkit. We use $N = 5$ for language modeling in SRILM. SRILM and Moses were used with the default options, except that the distinct option was also given to Moses to avoid any duplicates in the output list.

In the development phase, the performance of the baseline systems was improved by tuning the various model parameters. In case of CRF, we varied the C parameter which controls the trade-off between train and test error, and in case of SMT based transliteration, we tune the weights assigned to various models like language model and translation model. We disabled the distortion model by assigning it zero weightage. Note that, as discussed in Section 5.1, we use same enlarged alphabet for our system as well as for the baseline systems.

Having discussed the related work and evaluation criteria, we next give a step by step development of our system architecture.

4. TRANSLITERATION SYSTEM ARCHITECTURE

The main modules in our system are shown in Figure 1. To help motivate various design choices made by us, we present a detailed development history of our system. We start by presenting a very basic rule based system in Section 4.1. This system relies on manually created character mappings for generating transliteration candidates and uses a language model called Character Sequence Model (CSM) to rank the generated candidates. This rudimentary system is enhanced in Section 5 by fine-tuning various CSM parameters such as the character set being used, the order of the language model, the weight being assigned to each word, and the smoothing techniques being used.

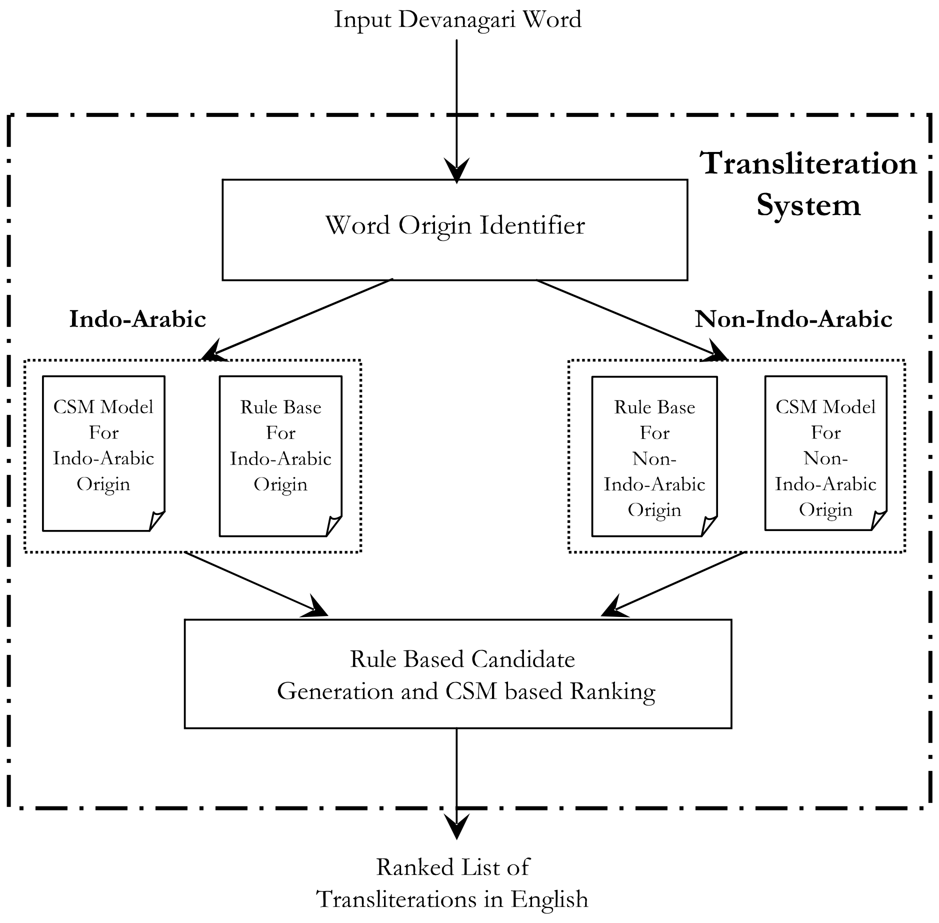
Input Devanagari Word
Transliteration System
Word Origin Identifier
Indo-Arabic
Non-Indo-Arabic
CSM Model For Indo-Arabic Origin
Rule Base For Indo-Arabic Origin
Rule Base For Non-Indo-Arabic Origin
CSM Model For Non-Indo-Arabic Origin
Rule Based Candidate Generation and CSM based Ranking
Ranked List of Transliterations in English

Fig. 1. System architecture of rule-based transliteration.

The system is further developed by identifying the word-origin and employing the origin dependent rule bases and language models in Section 6. This final system is evaluated for Hindi-English and English-Hindi transliteration tasks on various datasets and the results are presented in Sections 7 and 8.

4.1 Basic Rule-Based System

Our basic rule-based transliteration system works by employing a set of character mapping or character sequence mapping rules between languages involved. We have used this system for Hindi to English, English to Hindi, and Persian to English transliteration tasks. We illustrate the system architecture in detail using Hindi to English transliteration as an example.

Hindi words are written in Devanagari script while English words are written in Roman script. When there is no confusion, we use the terms *Devanagari word* and *Hindi word* interchangeably. Each Devanagari consonant symbol that is not followed by a vowel represents that consonant plus an

Table III. A Snippet from the Hindi to English Constrained Rule Base

Hindi Character	English Mapping
क्	k,c,q,ck| !S,ch,lk| !S| AV
ख्	kh
ग्	g,gh
एक्स	x
द्	d, th
ई	i,e,ee,ea,ei,ey,ie,y,eigh,ai| !S| AC
प्	p
अ	a,e,o,u,ϵ

inherent *schwa vowel sound* अ [Shukla 2000]. For example, दीपक is represented as द्+ई+प्+अ+क्+अ. Note that the schwa vowel is not pronounced in certain contexts.

In our system, most rules are of the form (ख् $\rightarrow$ *kh, ck, k*) where a single Hindi character ख् is mapped to one or more English character sequences *kh, ck, k*. Some rules are of the form (एक्स$\rightarrow$ *x*) where a Hindi character sequence एक्स is mapped to a single English character *x*. The rules sometimes also include constraints which specify the context in which they are applicable like Start of a Word (S), Ending of a Word (E), After Vowel (AV), After Consonant (AC), etc. Negation of these constraints is also allowed by prefixing the constraint with !. A snippet of the constrained rule base is shown in Table III. Since the *schwa* vowel sound अ gets dropped in some contexts, we also add a ϵ (NULL) character mapping (i.e., no character appended during candidate generation) corresponding to *schwa*.

Combination of different mapping options for each character in a input Hindi word results in different transliteration candidates. For example consider the Hindi word दीपक (*deepak*). As discussed before, दीपक is processed as द्+ई+प्+अ+क्+अ. As shown in Table III, द्, ई, प्, अ, and क् have 2, 10, 1, 5, and 6 possible mappings, respectively. Hence a total of 2*10*1*5*6*5=3000 transliteration candidates should be considered (Examples: $d+ee+p+a+k+\epsilon$, $d+i+p+\epsilon+k+a$, $d+ee+p+u+k+\epsilon$, etc.).

One faces two basic issues in such a rule-based system.

(1) A procedure is needed to rank the generated candidates.
(2) A search strategy is needed to avoid exploring the exponential number of candidates. For example, there are around 933 million candidates for the word चेकोस्लोवाकिया (*Czechoslovakia*).

The ranking function that we use is the word generation probability based on Character Sequence Model (CSM) discussed next. To avoid processing exponential number of candidates, we process the input characters one at a time, and greedily prune the list of partially generated candidates by keeping only top *k* ranked candidates.

4.2 Character Sequence Modeling (CSM)

A language generally allows only certain spelling patterns in its word formations. For example, sound formations such as *hlad* and *mgla* are not natural in

English. The aim of Character Sequence Modeling is to learn the permissible patterns.

A *Character Sequence Model* for a language is a probability distribution over sequence of characters within a word. Let $W = < c_1, c_2, \ldots, c_m >$ be a word where c_i is the i^{th} character in the word. The probability of observing the word W in the corpus is defined as:

$$P(W) = \prod_{i=1}^{n} P(c_i \mid c_{i-1}, \ldots, c_{i-N}) \; (N^{th} \text{ Order Markov Assumption})$$

The very first step in any language modeling work is the decision of what to model, that is, which corpus to model. Most researchers used the unique word list from the training data or some named entity list from the Web as the corpus being modeled. We use English Wikipedia as our target corpus. The Wikipedia includes words and named entities from all parts of the world (5.6 million word forms in 2007, of which 1.8 million word forms had frequency greater than 2) and continues to grow exponentially. Therefore, it serves as a good general purpose corpus for transliterating into English. We train a trigram CSM on the unique words of Wikipedia using the SRILM toolkit [Stolcke 2002] with the default options.

To summarize, the rule-based transliteration system uses a simple constrained rule base to generate all possible candidates. At each step, the candidates are ranked based on the word generation probability given by the CSM. To avoid processing exponential number of candidates, only the top k candidates are retained at each step and the rest are discarded.

5. IMPROVING THE BASIC RULE-BASED SYSTEM

Table IV shows the performance of our basic system and that of the baseline systems. Both the baseline systems employing parallel corpus outperform our basic system with a huge margin.

5.1 Enlarging the Alphabet on Target Side

In our system, the candidates generated are of varying length since one Hindi character can map to multiple English characters, and the CSM assigns lower probability to the longer candidates than to the shorter ones. For example, for word फाटक (*phatak*), candidate *fatak* is assigned higher probability compared to *phatak* because letter p and h are treated as two different graphemes, even though both *f* and *ph* correspond to the same Hindi letter फ. We solve this problem by changing the *alphabet* (basic unit) in our N-Gram model. We treat several multi-character sequences as a single character. In Ekbal et al. [2006], the basic alphabet is extended by extracting multi-character sequences derived from an alignment of parallel corpus. Since we are not using parallel corpora, we add multi-characters gram based on the following.

—Common Digraphs. As in Teahan [1997], we treat frequently occurring digraphs such as *sh*, *ck*, etc., as a single character.

Table IV. Hindi to English Development Set Results on 30,000 Dataset

The table shows the sequence of techniques used for improving the CSM performance along with the actual accuracy improvements observed. The performance of baseline CRF and SMT-based approaches is also given. In-vocabulary and out-of-vocabulary (OOV) accuracies are not reported for CRF and SMT since they do not use any additional monolingual resources.

30,000 Dataset	**Accuracy (%)**					
	Category-wise					
	Indo Arabic	**Non-Indo Arabic**	**In-Vocab**	**OOV**	**Overall**	**MRR**
Basic System (N=3)	29.4	33.6	32.9	20.3	31.3	0.187
Enlarge Alphabet (N=3)	33.9	39.4	38.4	20.9	36.4	0.222
Varying Order of CSM (N=5)	44.3	63.3	56.9	21.0	53.1	0.389
Weighing Each Word + Default Smoothing (GT Disc. + Katz BO)	52.4	71.7	66.3	17.3	61.4	0.454
CG Disc. + Kneser Ney Smoothing	52.6	72.7	66.7	20.3	61.9	0.461
PPM-D Smoothing	54.3	72.8	68.5	11.2	62.7	0.470
MLE (No Smoothing)	40.3	70.3	59.5	8.4	54.4	0.407
Common Rules + Diff. CSM + PPM-D	71.9	72.3	75.0	49.0	71.7	0.563
Diff. Rules + Diff. CSM + PPM-D	76.9	73.6	76.9	59.6	**75.1**	**0.612**
Baseline Approaches						
CRF Transliteration	81.1	49.8	-	-	66.6	0.525
SMT Transliteration	79.2	63.4	-	-	**71.9**	**0.548**

—Double letters. Typically, a pair of identical letters is pronounced as a single phoneme in English, and hence we add them to our alphabet. For example, *ll* and *ss* in *mill* and *miss*.

—Schwa handling. In Section 4.1, we discussed that in written words, each Devanagari consonant that is not followed by a vowel contains an inherent schwa vowel which may or may not be pronounced. For a simple rule-base system, it is hard to decide when a schwa will be pronounced and when will it be deleted. Hence we treat basic consonants followed by schwa to be a single letter for example, *ka, kha, ga*, etc.

As shown in Table IV, this enlarging of alphabet improves the performance of system from 31.3% to 36.4%. As an example, candidates for word इम्प्रेस (*impress*) changes from {*impres*, *empres*, *empras*, *imprec*} in the basic system to {*impres*, *empres*, ***impress***, *empress*, *impras*}.

5.2 Varying N of N-Gram

So far we have been using the trigram model because that is the default in the SRILM toolkit, and it is the model used by many other transliteration researchers. We next vary the value of N as shown in Figure 2 and find that $N = 5$ gives the best results and this increases the accuracy from 36.4% to 53.1%. For example, candidates for word डिजॉल्व (*dissolve*) changes from {*dicallo*, *dicalo*,

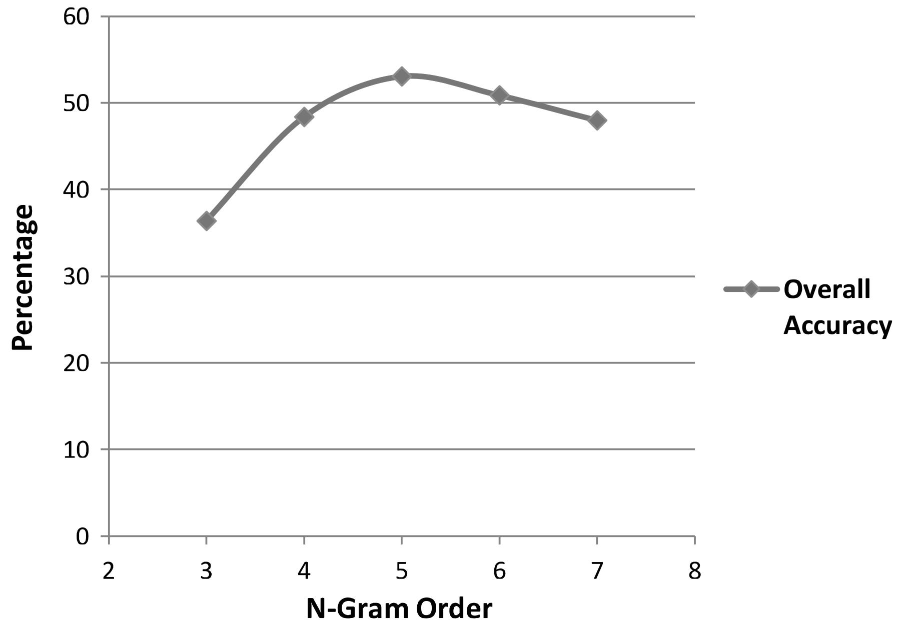

Fig. 2. Graph showing the effect of CSM order on transliteration accuracy at rank 5 on 30,000 dataset development set.

decolo, *disolo*, *dicalva*$\}$ to $\{$*dissolo*, ***dissolve***, *decolo*, *disallo*, *desalu*$\}$ taking benefit of *d i ss o l* as a 5-gram unit in the training corpus.

5.3 Weighing Each Word

Until now, we have been using the list of unique words from the Wikipedia for learning the Character Sequence Model on the target side. A list of unique words gives equal importance to all the words in the corpus, whether it be a correct spelling, or a misspelling, or a rare spelling variation. To give more importance to the frequent spellings, we could use word frequency as the weight of a word when modeling the CSM. Given that many words occur with very high frequency, such a model will unduly favor the common words. To strike a balance, we use the following weighting function for a word w when learning the CSM model:

$$wt(w) \;=\; \lfloor ln(f(w)) \rfloor.$$

Here, $f(w)$ is the frequency of word w in the corpus. As a desirable side effect, words occurring only once or twice are not used at all and the number of unique word forms in the corpus reduces from 5.6 million to 1.8 million. This weighing increases the system accuracy from 53.1% to 61.4%. For example, candidates for डैनिश (*danish*) changes from $\{$*dennis*, *danes*, *denis*, *denes*, *danese*$\}$ to $\{$*denis*, ***danish***, *danes*, *dennis*, *danese*$\}$.

5.4 Smoothing and Discounting Techniques

The experiments so far have used the default SRILM smoothing, which is the Katz Backoff with Good-Turing Discounting. While this is known to work well for word sequence modeling tasks, Character Sequence Modeling may require a different smoothing method. The number of characters in a given alphabet is orders of magnitude smaller than the number of words in a language. Hence the number of N-Grams to be considered in a transliteration application is far less than that in a word modeling task. Also, the number of distinct sentences a system may have to process is infinite, while the number of possible words is arguably finite. Therefore, the smoothing technique for the CSM should give less weightage to the unseen N-Grams than what is typically done in the word modeling tasks.

We compare the performance of Katz Backoff and Good-Turing Discounting with two other methods: Chen-Goodman Backoff with Kneser-Ney Discounting (implemented in the SRILM toolkit), and PPM-D smoothing (not implemented in SRILM) [Teahan 1997]. Since methods implemented in SRILM toolkit are widely known in NLP community, we do not describe them here. We only explain the PPM-D smoothing which is relatively unknown in the NLP community.

Prefix-Based Partial Match (PPM) is an adaptive N-gram character sequence model well-known in the text compression community [Cleary et al. 1984]. Let a be the context observed so far and z be the next symbol in sequence. Let $c(z)$ be the number of times the context a was followed by the symbol z and n ($\sum_k c(k)$) be the total number of symbols that have followed a; Let t be the number of distinct symbols that have followed context a so far and M be the total number of distinct symbols seen in the training data. We use the PPM-D smoothing method [Teahan 1997] which is given by:

$$Pr(z \mid a) = \begin{cases} \frac{2 \cdot c(z) - 1}{2 \cdot n} & \text{If } c(z) > 0 \text{ in context } a \\ \frac{1}{M-t} \cdot \frac{t}{2 \cdot n} & \text{Otherwise} \end{cases}$$

Since we want to give much less weight to the unseen N-Grams, a natural baseline method is to use Maximum Likelihood Estimate (MLE), and employ no smoothing. We also present the comparison results for this option.

The results of applying various smoothing techniques are shown in Table IV. To put the results in perspective, we categorized them by whether the correct English transliteration is present in the corpus being modeled or not. The Kneser-Ney technique gives the best result for the out-of-vocabulary (words which are not in Wikipedia) target words while the PPM-D smoothing gives the best result for the in-vocabulary words. The performance of the PPM-D is not unexpected since it concentrates the probability mass heavily on the seen events. The choice of the smoothing technique depends on whether we expect to see lot of out-of-vocabulary words in our application or not. Hence for a general-purpose system with Wikipedia as the target corpus, we expect to run into the in-vocabulary words much more often than the out-of-vocabulary words. Therefore, we chose PPM-D smoothing. With this choice, a seen word like कुशीनगर (kushinagar) breaks into the top 5 with the candidate set changing from {*choosinger*, *cusinger*, *cussinger*, *cousinger*, *kusinger*} to {***kushinagar***, *kusinagara*, *cousinger*, *kuchinger*, *cusinger*} due to the less weightage given to unseen sequences like *choosinger*, *cusinger* etc.

6. WORD ORIGIN IDENTIFICATION

It is observed in Huang [2005] and Surana and Singh [2008] that identifying the word origin is critical to the transliteration success. In Huang [2005], a statistical clustering technique is employed on a parallel transliteration corpus, and 50 different classes of English words are created. Since our work focuses on the use of monolingual resources, ideally we should classify words based on the phonological typology. In the absence of resources to do such a classification, we simply classify words by whether they are of Indian origin or not. In Shukla [2000] the origin of Hindi words is traced to four sources: a) Native words (Sanskrit words and their derivatives), b) Persian, Arabic, and Turkish words, c) English words, and d) Portuguese words. Since not only the words of other Indian languages, but even the words of Persian (e.g., ज़मीन *zameen*), Arabic (e.g., औरत *aurat*), and Turkish (e.g., दरोगा *daroga*) sound like Hindi words to native speakers, we finally classified words by whether they are of Indo-Arabic origin or not. Note that in the context of resource scarce languages, effort required to manually annotate the origin of a given word is much less than that of producing the correct transliteration.

Previously, *N*-Gram models have been used for the word origin identification in Llitjos and Black [2001] and Surana and Singh [2008]. We use a similar model where we learn two different CSM classes corresponding to each origin and assign a word to the class which gives it higher probability. The 30,000 and NEWS 2009 datasets were manually annotated with word origin information. The word origin data of training set was used for training the word origin identification module. The development set was used to determine the ideal *N*-Gram size. The *N*-Gram size 3 gives the best accuracy of 87% for the word origin identification task. In Surana and Singh [2008], 5-grams are found to be most effective for identifying the origin of words written in the Roman script. In contrast, we find that 3-grams are best suited for the origin identification of written Hindi words. This is because many consonants in written Devanagari words include an inherent schwa vowel, as discussed in Section 4.1.

6.1 Origin-Dependent CSM

The purpose of word origin identification is to use a different origin-dependent CSM on the target side. For training the English CSM for the words of Indo-Arabic origin, a collection of 167,814 Indo-Arabic names written in the Roman script were used. For non-Indo-Arabic origin words, the English Wikipedia corpus was used. While Wikipedia also contains many words of Indo-Arabic origin, words of Non-Indo-Arabic origin are orders of magnitude more frequent.

The system works as before, except that the character sequence model for ranking the candidates is now based on the origin assigned by the classifier as depicted in Figure 1. As shown in Table IV, the accuracy of the system now jumps from 62.7% to 71.7%, exceeding that of the CRF. Note the particular increase in the accuracy of Indo-Arabic origin words, since the language model for them gets quite altered. As an example, बोधगया (*bodhgaya*) benefits from the changed CSM where the prefix *b o dh* gets preference over the prefix *b o d* and the candidates change from {*bodgaya*, *bodgia*, *bodegua*, *bodeguo*, *bodegea*} to {*bodhagaya*, *bodhagya*, *bodhgya*, ***bodhgaya***, *bodhegya*}. For non-Indo-Arabic origin words the CSM has not really changed much. Compared to results shown in Table IV, the slight reduction in the accuracy of foreign origin words is due to misclassification errors; it is fatal for a non Indo-Arabic word to be classified as that of Indo-Arabic origin. For example, when सन्निवेल (*sunnyvale*) gets misclassified as Indo-Arabic origin, its candidates change from {***sunnyvale***, *snivel*, *snivell*, *snivelle*, *conewall*} to {*sanewal*, *shanewal*, *sanewala*, *shanewala*, *shaneevala*}.

6.2 Origin-Dependent Rule Base

The error analysis at this stage still shows scope for improvement. Regardless of the word origin, same set of candidates are currently being generated. Hence we decided to use different character mapping depending on the word origin. This step is arguably tricky. We cannot characterize our system as a simple one anymore. In fact, the mapping rules were improved using trial-and-error. Despite having these misgivings, we think that such a step still helps us show the power of CSM. The system guesses the origin of the input Hindi word, generates candidates from the appropriate mapping table based on the word origin, and ranks the generated English candidates using the appropriate CSM. The final system architecture is shown in Figure 1. The performance of this system is shown in Table IV. From 71.7%, the system accuracy increases to 75.1% compared to 71.9% accuracy of the SMT based system. Its performance is comparable to that of the SMT-based system. The major beneficiaries of Origin Dependent Rule Base are words of Indo-Arabic origin. For example, क् maps to only *k* and not to *ch* in the Indo-Arabic origin rule base and hence the candidates for कामराव (*kamrao*) changes from {*chamaroo*, *kamaroo*, *kamarava*, *chamarava*, *komrao*} to {*kamarava*, ***kamrao***, *kamarav*, *kaamrao*, *kamerava*}.

7. EXPERIMENTAL RESULTS AND ERROR ANALYSIS

Until now we have been optimized our system based on its performance on the development data. We next experiment with the test set of 30,000 dataset mentioned in Table II. As mentioned before, a five-fold cross validation was performed on this dataset and all reported numbers are the average figures for five runs. We repeated the same sequence of steps on the NEWS 2009 dataset.

The results on the two datasets are given in Tables V and VI, and Figure 3. From these results, we can conclude that our system competes fairly well with statistical systems for Hindi-English transliteration task. Our chosen baselines do not take advantage of the word origin identification. To remedy this, we also implemented an improved version of the baseline systems where the training data was separated by word origins. The results of that experiment on NEWS 2009 dataset is shown in Table VII. Surprisingly, the overall result

Table V. Hindi to English Test Set Results on 30,000 Dataset

Development set results repeated for case of comparison.

30,000 Dataset	Accuracy (%)			
	Category-wise			
	Indo Arabic	Non-Indo Arabic	Overall	MRR
Dev				
CRF Transliteration	81.1	49.8	66.6	0.525
SMT Transliteration	79.2	63.4	71.9	0.548
Diff. Rules + Diff. CSM + PPM-D	76.9	73.6	**75.1**	**0.612**
Test				
CRF Transliteration	79.2	49.4	65.8	0.516
SMT Transliteration	77.9	64.4	70.6	0.532
Diff. Rules + Diff. CSM + PPM-D	75.3	73.2	**73.1**	**0.591**

Table VI. Hindi to English Development and Test Set Results on NEWS 2009 Dataset

Development set results repeated for ease of comparison.

NEWS 2009	Accuracy (%)			
	Category-wise			
	Indo Arabic	Non-Indo Arabic	Overall	MRR
Dev				
CRF Transliteration	80.2	50.8	69.2	0.550
SMT Transliteration	79.2	69.0	**75.3**	**0.594**
Diff. Rules + Diff. CSM + PPM-D	70.4	69.6	70.1	0.556
Test				
CRF Transliteration	77.5	52.9	71.2	**0.550**
SMT Transliteration	73.2	62.9	**71.6**	0.520
Diff. Rules + Diff. CSM + PPM-D	68.7	61.0	69.8	0.547

for statistical systems does not benefit from the word origin identification. Our preliminary investigations indicate that the effect of origin misclassification is much more detrimental for statistical systems than the rule-based systems but we need to further investigate the root causes for this.

7.1 Error Analysis

There are several sources of error in our system.

—*Multiple Transliterations*. For many words of non-Indian origin, multiple English words correspond to a given Hindi word and our system fails to guess the one in the Gold Standard. For example, for गेट्स (*gets, gates*), top five results from our system are: {*gets*, *ghats*, *getz*, *geths*, *geats*}, whereas the gold standard only contains *gates*.

—*Origin Misclassification.* As discussed in Section 6.1, origin misclassification is one of the important sources of error.

—*Lack of Context Sensitive Mapping Rules*. While the previous two causes of errors are system errors, this one is a model error. Only context we use is whether a character is at the beginning of a word or at the end of a word, and whether or not it follows a vowel. As a result our top five candidates for शेयर (*share*) are *chair*, *cheer*, *seer*, *sheer*, and *sauer*, whereas a systems with rules like श followed by a य gets transliterated as *sh* only might give the correct answer.

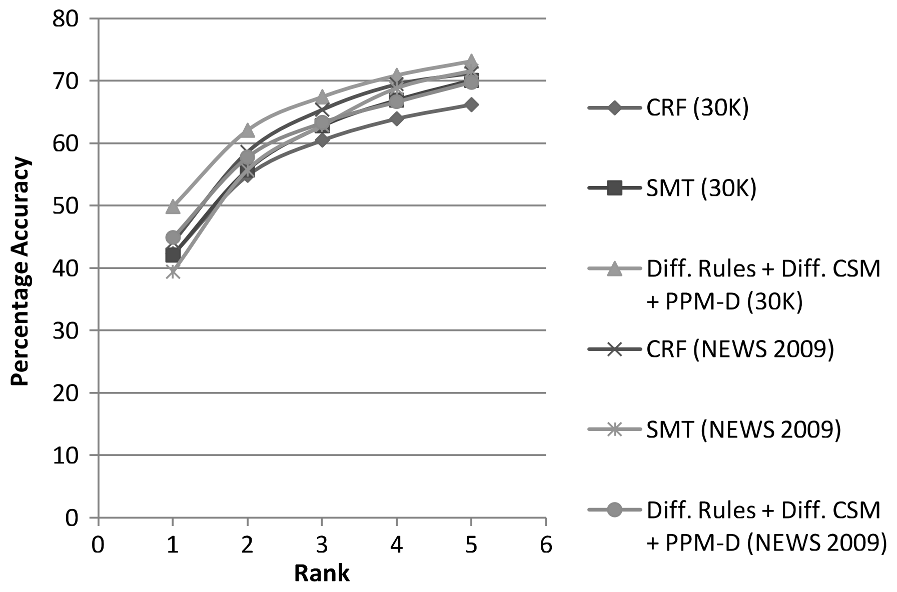

Fig. 3. Hindi-to-English test set rank vs. accuracy.

Table VII. Effect of Origin-Wise Training of Baseline CRF and SMT Approaches on NEWS 2009 Dataset

NEWS 2009 Test	**Accuracy (%)**			
	Category-wise			
	Indo Arabic	**Non-IndoArabic**	**Overall**	**MRR**
Hindi-English				
CRF	77.5	52.9	71.2	0.550
CRF + Word Origin	75.8	56.2	70.7	**0.556**
SMT	73.2	62.9	71.6	0.520
SMT + Word Origin	75.2	61.0	**72.1**	0.516
Diff.Rules+ Diff. CSM + PPM-D	68.7	61.0	69.8	0.547
English-Hindi				
CRF	68.4	56.8	65.4	0.473
CRF + Word Origin	69.2	60.7	**67.0**	0.492
SMT	67.5	55.2	64.4	0.483
SMT + Word Origin	67.7	54.4	64.4	**0.494**
Diff.Rules+ Diff. CSM + PPM-D	69.6	35.7	59.2	0.459

—*CSM Related.* In a system based on language modeling on target side, more prevalent spelling patterns in the target language are ranked higher than less prevalent spelling patterns.

—*Schwa Related Errors.* Our system cannot distinguish between different role played by letter ड in two very closely related words like धडकन (*dhadkan*) and धडक (*dhadak*) due to schwa deletion phenomena discussed earlier. Although the context of letter ड (*da*) is same in both cases, the schwa is dropped in one case and not in another.

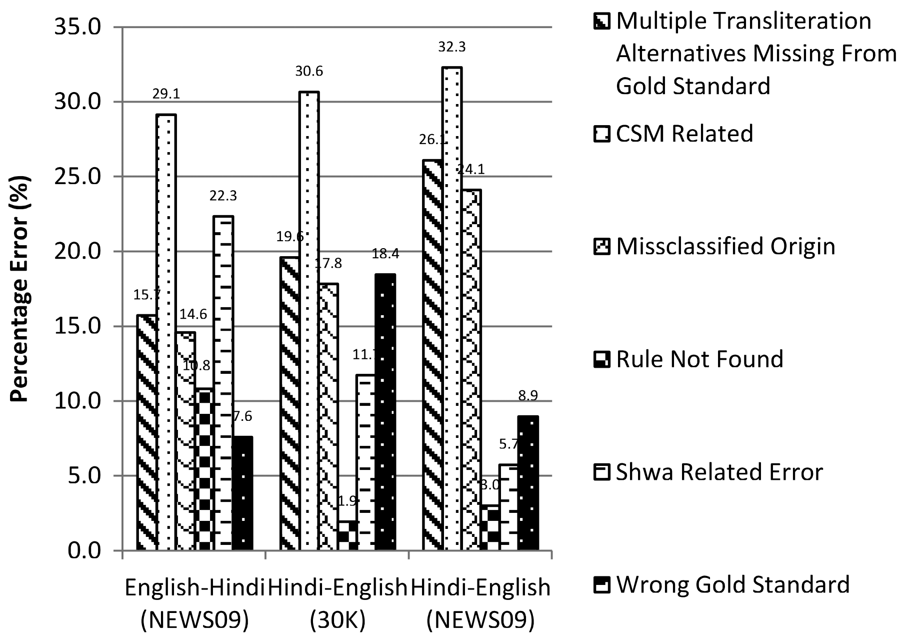

Fig. 4. Error analysis on Hindi-English and English-Hindi datasets.

The percentage of errors belonging to various categories is shown in Figure 4. For comparison sake, this figure includes the error distribution for English-Hindi transliteration task described next.

8. ENGLISH TO HINDI TRANSLITERATION

In the previous sections, we described our approach for transliterating from a phonetic script (Devanagari) to a non-phonetic script (Roman). In this section, we show that our approach works in the reverse direction as well, that is, it works for transliterating from a non-phonetic script (Roman) to a phonetic script (Devanagari).

8.1 English-to-Hindi Rule Base

As done earlier, we build a constrained rule-base for mapping English character sequences to Hindi character sequences. Similar to Hindi to English, the constrained rule-base contains both simple and compound rules where a rule is defined as *simple* or *compound* depending on whether the source of a rule is a single English letter ($k \rightarrow$ क) or an English letter sequence ($ck \rightarrow$ क). In English, vowel sequences (Example: *oe, eo, ie, ee, ai*) and silent letters (e.g., *lm (holmes), ps (pseudo))* are handled through compound rules.

While transliterating from English to Hindi, due to the Schwa phenomenon discussed in Section 5.1, English vowels may not be mapped to any character on the Hindi side. For example, in *ganesh* $\rightarrow$ गणेश, *but* $\rightarrow$ बट, *ton* $\rightarrow$ टन, the letters *a, u, o* do not map to any Hindi character. To handle this, we include NULL mapping on the target side for each English vowel. For example, both

बुट (contains three characters) and बट (contains two characters) are generated as transliteration candidates for *but*. Since the shorter candidates are likely to have higher probability in CSM, the NULL mapping may end up getting preference over other mappings. As discussed in Section 5.1, we take care of this by enlarging the target side alphabet. We fuse the Hindi vowels with their previous consonants and add them to the alphabet. As a result, the NULL and non-NULL mappings for English vowels contribute the same number of character units towards the generated transliteration candidates thereby solving the length bias problem. For example, as बु is added to the alphabet, both बुट and बट contain two character units only.

8.2 Experiments and Results

For evaluation of English-to-Hindi transliteration, we use the standard NEWS 2009 shared task dataset, the details of which are already given in Table II. We use the training set to train the CRF and SMT transliteration systems used for the baseline comparisons. The development set was used to tune the various CSM parameters in our rule-based system and other parameters in the baseline statistical systems.

For training the CSM in Hindi, we make use of a Hindi Web crawl of 427,067 documents obtained from *Guruji*,[1] an Indian Search Engine company. We extract the list of unique Hindi words from the above corpus and train a basic tri-gram CSM using SRILM toolkit. Later, we progressively improve the CSM accuracy by following the sequence of techniques mentioned previously like enlarging alphabet, varying CSM order, varying smoothing technique, using origin-wise rule base and CSM. The results on the development set is shown in Table VIII. The origin classification module was trained on English side using the same training data as mentioned in Table II.

The results on the test set are shown in Table IX and Figure 5. Unlike the Hindi to English transliteration results shown in Table V, the accuracy of the rule-based system is lower than both CRF and SMT approaches. However, its performance for the words of Indo-Arabic origin still remains respectable. Since written English is non-phonetic, pronunciation of a letter depends lot more on the context, and hence a rule based system that does not pay much attention to context fares poorly. A detailed comparison of the relative complexities involved in Hindi to English and English to Hindi transliteration tasks will be presented in Section 9.

9. ANALYSIS OF TRANSLITERATION COMPLEXITY

In this section, we analyze the results of the rule-based system using an entropy-based measure which allows the comparison of transliteration complexity across various datasets. For the rule-based system, we

—compare the relative complexity of Hindi-English and English-Hindi transliteration tasks; and

[1]See http://www.guruji.com.

Table VIII. English to Hindi Development Set Results on the NEWS 2009 Dataset

NEWS 2009 Dataset	Accuracy (%)					
	Category-wise					
	Indo Arabic	Non-Indo Arabic	In-Vocab	OOV	Overall	MRR
Basic System (N=3)	49.7	22.3	43.5	18.7	37.2	0.220
Enlarge Alphabet (N=3)	66.3	39.9	65.2	21.9	54.2	0.378
Varying Order of CSM (N=4)	65.6	43.7	67.1	21.6	55.5	0.410
Weighing Each Word + Default Smoothing (GT Disc. + Katz BO)	70.4	47.0	73.5	19.7	59.7	0.471
CG Disc. + KneserNey	69.7	44.5	71.1	21.2	58.3	0.427
PPM-D	68.7	47.8	75.6	11.3	59.2	**0.488**
MLE	46.1	29.5	51.5	0.6	38.5	0.331
Common Rules + Diff. CSM + PPM-D	68.2	45.5	73.2	13.6	57.9	0.452
Diff. Rules + Diff. CSM + PPM-D	73.9	44.7	73.3	24.3	**60.7**	0.481
Baseline Approaches						
CRFTransliteration	70.5	52.0	-	-	63.3	0.463
SMTTransliteration	73.4	64.0	-	-	**69.8**	**0.546**

Table IX. English to Hindi Test Set Results on the NEWS 2009 Dataset

NEWS 2009	Accuracy (%)			
	Category-wise			
	Indo Arabic	Non-IndoArabic	Overall	MRR
	Dev			
CRF Transliteration	70.5	52.0	63.3	0.463
SMT Transliteration	73.4	64.0	**69.8**	**0.546**
Diff. Rules + Diff. CSM + PPM-D	73.9	44.7	60.7	0.481
Test				
CRF Transliteration	68.4	56.8	**65.4**	0.473
SMT Transliteration	67.5	55.2	64.4	**0.483**
Diff. Rules + Diff. CSM + PPM-D	69.6	35.7	59.2	0.459

—analyze the effect of dataset choice on transliteration accuracy for Hindi-English transliteration.

9.1 Average Entropy of a Dataset

Let L_1 and L_2 be two languages with alphabets $A_1 = \{x_1, x_2, \ldots, x_m\}$ and $A_2 = \{y_1, y_2, \ldots, y_n\}$. For the transliteration task $L_1 \rightarrow L_2$, given a dataset D with parallel list of transliterations, we measure the uncertainty involved in mapping characters of L_1 to characters of L_2 using *Average Entropy* [Jurafsky and Martin 2008] defined as follows:

$$\begin{aligned} AvgEntropy(L_1, L_2, D) &= \frac{\sum_{x_i \in A_1} Entropy(x_i)}{\mid A_1 \mid} \\ &= -\frac{\sum_{x_i \in A_1} \sum_{y_j \in A_2} Pr(y_j|x_i) \cdot logPr(y_j|x_i)}{\mid A_1 \mid} \end{aligned} \quad (1)$$

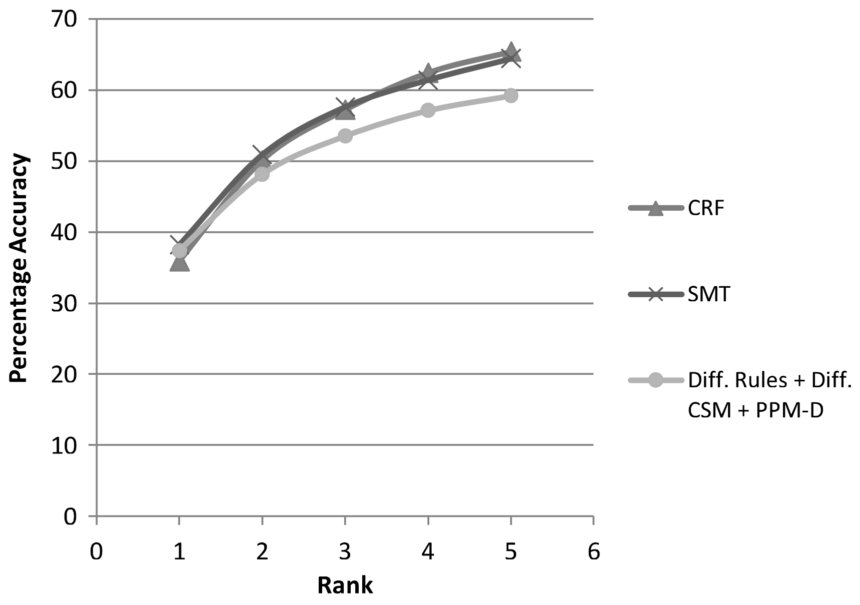

Fig. 5. English to Hindi test set rank vs. accuracy on the NEWS 2009 dataset.

Here, $Pr(y_j|x_i)$ represents the conditional probability of observing the target character y_j corresponding to the source character sequence x_i in the given dataset D.

Characters which have a deterministic mapping in the target language (e.g., ख् $\rightarrow$ kh) have zero entropy and do not contribute to the overall entropy whereas characters whose mappings are highly ambiguous and context-dependent (e.g., क् $\rightarrow$ k,c,q,ck|!S,ch) have high entropy and contribute more. Hence, $AvgEntropy(L_1, L_2, D)$ serves as an indicator for the relative hardness of the transliteration task for the language pair $L_1 \rightarrow L_2$ on a given dataset D.

Given the dataset D, we run GIZA++ aligner from $L_1 \rightarrow L_2$ to obtain the character-level alignments and compute $AvgEntropy(L_1, L_2, D)$ using Equation 1.

9.2 Hindi-to-English vs. English-to-Hindi Transliteration

Intuitively, we expect the transliteration from a non-phonetic language to a phonetic language to be harder than the reverse task. We use the *AvgEntropy* measure defined above to formalise the above intuition. While calculating *AvgEntropy*, we use the same NEWS 2009 dataset for both directions. The result is shown in Figure 6 which correlates the accuracy of the rule-based system with the average entropy.

The results show that the average entropy of Hindi to English is less than English to Hindi and hence the task of Hindi-to-English transliteration is expected to be easier when compared to English-to-Hindi transliteration. Also, we find that the uncertainty involved in character mapping is most significant in case of English vowels. This is not surprising since there are many more

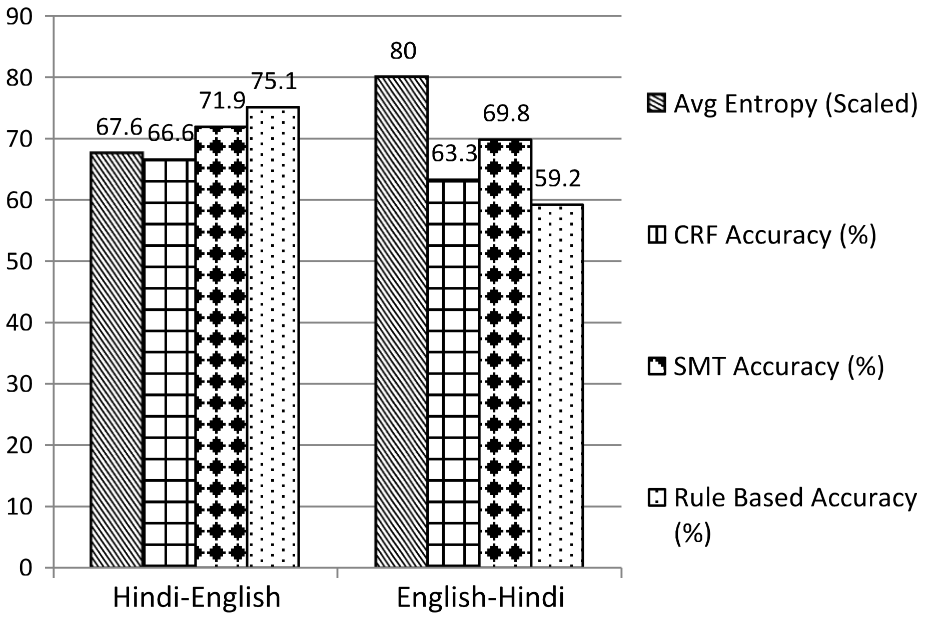

Fig. 6. Average entropy vs. rule based system accuracy for Hindi to English and English to Hindi transliteration.

vowel sounds in English compared to only the six vowels in the roman alphabet especially if dipthongs are treated as a single unit [Rollings 2004].

9.3 Effect of Dataset Choice on Transliteration Accuracy

The results reported on the test set in Table V do not outperform the best accuracy numbers reported in transliteration literature. Currently researchers report widely varying accuracy numbers: from 32%-88% [Karimi et al. 2007] at Rank 1. For Hindi-to-English transliteration, Ganesh et al. [2008] report 72.1% accuracy at rank 5 for a CRF based system, while Kumaran and Kellner [2007] reported 31.1% accuracy at Rank 10. An important factor affecting any transliteration system performance is the quality of the test set, as discussed in Karimi et al. [2007]. On a different parallel name list of 5,000 Indian names (will be referred to as Indian Names Test (5,000)), our system gives accuracy of 87.4% at Rank 5. But this list seems to have been generated using a simple mapping table and is not representative of the real-life spelling variations and idiosyncrasies, and does not have enough instances of the *schwa deletion* phenomenon. Hence, it is highly imperative that results reported on non-open or non-standard datasets[2] should also report measures similar to average entropy to quantify the inherent hardness of transliteration.

Ideally, we expect a system's performance to degrade as the entropy of the test set increases. In Figure 7, the average entropy and the Hindi-to-English

[2]Datasets used in this article are publicly made available at http://www.cse.iitb.ac.in/˜damani/papers/TALIP10/data.tgz.

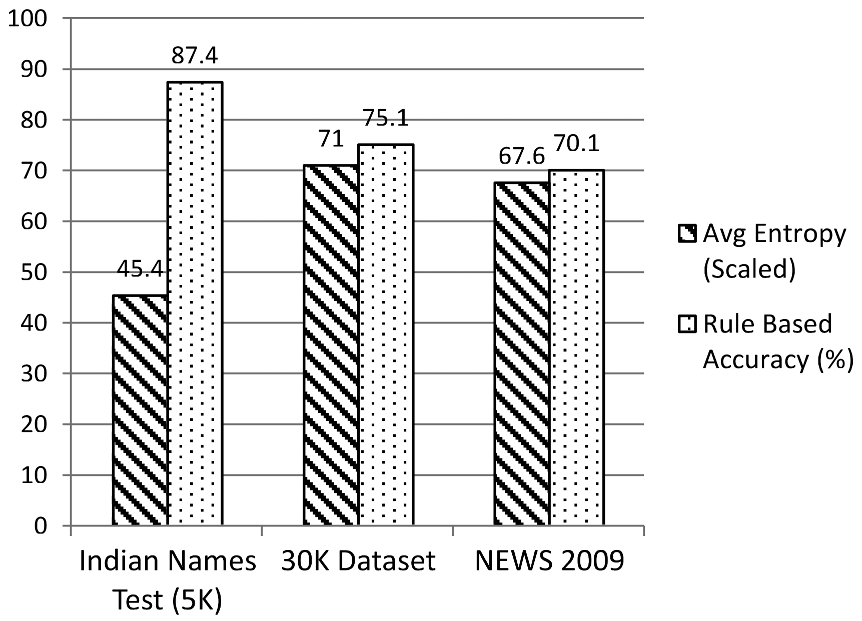

Fig. 7. Average Entropy vs. Rule-Based System Accuracy for Hindi to English across different datasets.

transliteration accuracy is shown for the different dataset 30,000, NEWS 2009 and Indian Names Test (5,000). As discussed earlier, Indian Names Test (5,000) is the simplest of all three and hence has the least average entropy. The 30,000 dataset is the hardest, followed by NEWS 2009. However, the results reported in Tables V and VI are not in sync with the above observation. Indian Names Test (5,000) has the highest accuracy. But NEWS 2009 has lower accuracy than 30,000 which has higher entropy. This shows that Average Entropy alone cannot fully explain the performance of a system. We next provide a cross entropy based explanation for the above trend.

9.3.1 *Cross Entropy of Rule-Based System on Datasets.* Our non-probabilistic rule based system has rules of the form $x \rightarrow \{y_1, y_2, \ldots, y_k\}$ listing all possible target language character mappings corresponding to a letter x in source language. During candidate generation, all the target language character mappings $\{y_1, y_2, \ldots, y_k\}$ are treated as being equally probable and hence the conditional probability distribution $Pr_{RB}(y|x)$ used by the rule base could be stated as:

$$Pr_{RB}(y|x) = \begin{cases} \frac{1}{k} & \text{If } y \in \{y_1, y_2, \ldots, y_k\} \\ 0 & \text{Otherwise} \end{cases}$$

Hence, in our rule-based system, the actual conditional probability distribution $Pr_D(y|x)\}$ associated with a dataset D is approximated by $Pr_{RB}(y|x)$. The

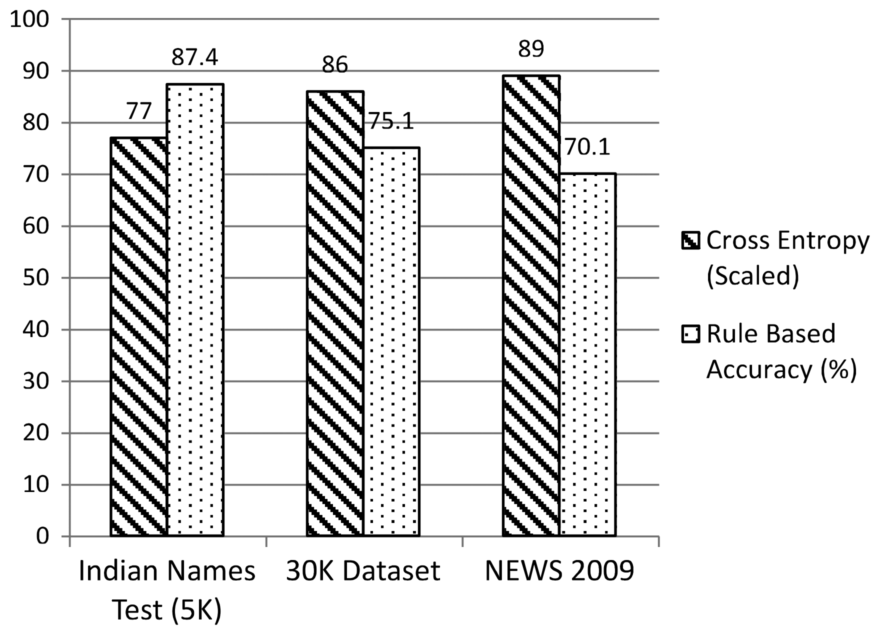

Fig. 8. Average Cross Entropy vs. Rule-Based System Accuracy for Hindi to English across different datasets.

Average Cross Entropy [Jurafsky and Martin 2008] of Pr_D with respect to Pr_{RB} given by:

$$\begin{aligned} AvgCrossEntropy(L1, L2, D, RB) &= \frac{\sum_{x_i \in A_1} CrossEntropy(x_i, D, RB)}{\mid A_1 \mid} \\ &= -\frac{\sum_{x_i \in A_1} \sum_{y_j \in A_2} Pr_D(y_j|x_i) \cdot log\ Pr_{RB}(y_j|x_i)}{\mid A_1 \mid} \end{aligned} \tag{2}$$

$AvgCrossEntropy(L1, L2, D, RB)$ captures the relative hardness of the transliteration task on the dataset D when the rule-based system $Pr_{RB}(y|x)$ is used to approximate the actual distribution $Pr_D(y|x)$.

The average cross entropy of the three Hindi to English datasets and the corresponding transliteration accuracies are shown in Figure 8. We observe that average cross entropy adequately explains the observed performance.

In the case of the rule-base system, mapping probabilities are assumed to be uniform, unlike the CRF and SMT based systems where these probabilities are learnt from a training set. The performance of these statistical systems will depend on the extent of similarity between their training and testing data. A cross-entropy measure can be similarly computed for each of these systems.

To summarize, it is important to understand the nature of training and testing data involved before comparing results obtained using different datasets.

Table X. FIRE 2008 Hindi and English Collection Statistics

Language	No. of Documents	No. of Unique Terms	Avg. Doc. Length	No. of Queries
English	125,586	191,769	265	50
Hindi	95,215	177,206	413	50

10. END-END CLIR EVALUATION USING FIRE 2008 DATASET

In the previous sections, transliteration approaches were evaluated *intrinsically*, as a independent standalone module. In this section, we evaluate the Hindi-to-English and English-to-Hindi transliteration approaches in the context of Cross Lingual Information Retrieval using the FIRE 2008[3] dataset. The details of the FIRE 2008 Hindi and English datasets are given in Table X. For CLIR, we use a *query translation* based CLIR approach using bilingual dictionaries [Padariya et al. 2008]. For the query words which are not found in the bilingual dictionary, their top five transliteration candidates are fed to the query translation engine. End-to-end cross lingual information retrieval was performed using each of the three transliteration approach (CRF, SMT, and Rule-Based). The standard method for CLIR evaluation is to compare the MAP (Mean Average Precision) score for cross-lingual evaluation with a monolingual baseline as % monolingual MAP. The results of our evaluation are shown in Figures 9 and 10. As expected, the effectiveness of the retrieval is directly proportional to the transliteration accuracy.

11. APPLICABILITY TO OTHER LANGUAGE PAIRS

Having described our system for Hindi-English and English-Hindi transliteration tasks, we now wish to explore the generalizability of our ideas. As per Frawley [1992] and Crystal [1992], a majority of the existing writing systems are *phonological* and have a clear relationship between sounds and symbols. The major exceptions are the *lopographic/morphographic* Chinese script and the Japanese Kanji script. Our approach to rule-based transliteration is applicable only to languages with phonological writing systems. Phonological systems are further divided into four categories: syllabic (e.g., Japanese Kana), only consonant letters (e.g., Arabic), independent consonant and vowel letters (e.g., Roman), independent vowel and integrated consonant-vowel letters (e.g., Indian Devanagari). Cutting across these categories, some writing systems are over-determined—they use multiple letters for one sound, while others are under determined—they use one letter for multiple sounds.

Using our rule-based system, it will be harder to transliterate into over determined writing systems, or to transliterate from under determined writing systems. In these scenarios, contextual information will be vital for disambiguation. Such information can be obtained either by a detailed understanding of the phonology of the language or by exploring a large parallel corpora. For all other settings, a rule-based approach may be applicable.

To verify our intuition, we decided to experiment with a non-Indian language. Our only criterion for selecting the language was that there should be

[3]See http://www.isical.ac.in/˜clia/2008/.

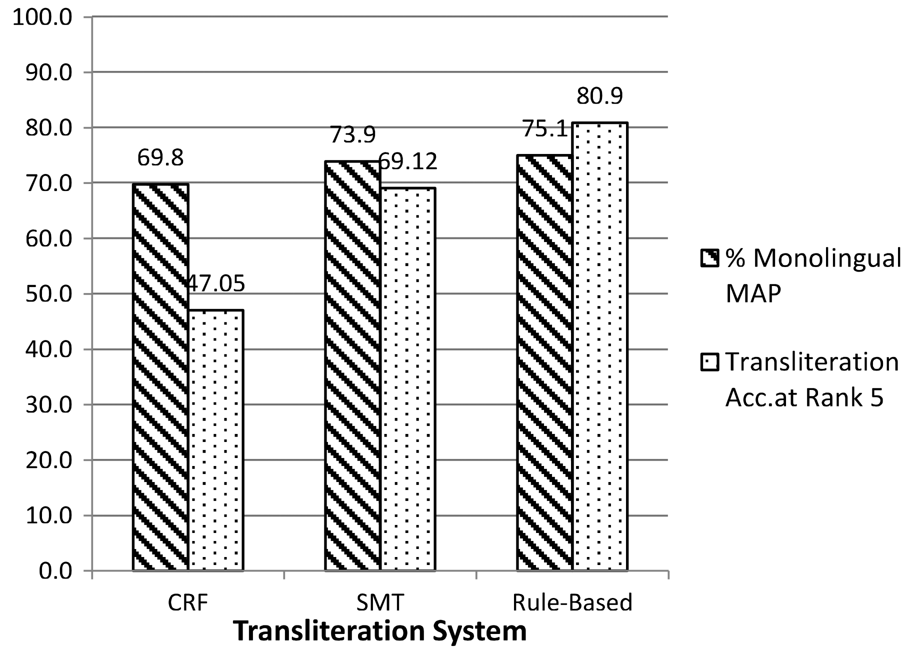

Fig. 9. FIRE 2008 Hindi to English CLIR dataset: Correspondence between transliteration accuracy and % monolingual MAP.

two publicly available resources; a character mapping into English, and a parallel transliteration corpus for evaluation. We next describe our experiments with Persian to English transliteration task. We had to spend only two days to get the basic experiments done, thus showing the adaptability of our approach.

11.1 Persian-to-English Transliteration

Character mappings and parallel transliteration corpus are publicly available for Persian to English transliteration in Karimi [2008]. Hence, we decided to work on it, despite the fact that none of the authors have any knowledge of the Persian whatsoever (we cannot even read the Persian script). Persian being a consonantal language, short vowels are typically absent from the written Persian words [Karimi 2008], making the problem even more challenging. The Persian-English parallel corpus of 19940 word pairs used in Karimi [2008] was randomly partitioned by us into training, development, and test set, details of which are given in Table XI. Due to the lack of data, we could not perform the extrinsic evaluation of Persian-to-English transliteration in the context of CLIR.

11.2 Persian-to-English Rule Base

We augmented the rule base obtained from Karimi [2008] slightly with the help of English to IPA (International Phonetic Alphabet) and Persian to IPA mappings. If an English character sequence and a Persian character sequence map

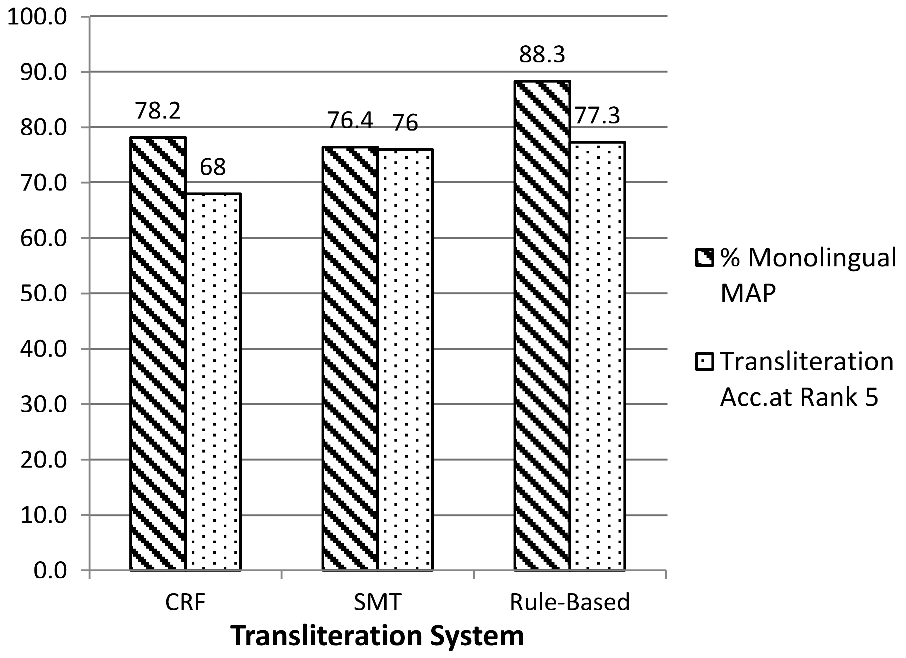

Fig. 10. FIRE 2008 English to Hindi CLIR dataset: Correspondence between transliteration accuracy and % monolingual MAP.

Table XI. Persian Dataset Details

	Perso-Arabic Origin	**Non Perso-Arabic Origin**	**Total (19940)**
Train	2,295	11,645	13,940
Dev	509	2,491	3,000
Test	483	2,517	3,000

Table XII. Persian to English Rule-Base Snippet

Persian Character	**English Mapping**
ا	a,aa,au
ف	f,ph
ی	y,ei,ay,ai,i,ee
س	c,s
ش	sh

to same IPA symbol, we mapped these sequences to each-other, for example, ی and *ei, ay, ai, ee*. A segment of the mapping rules is shown in the Table XII. Also, since short vowels are dropped from written Persian words, if a Persian consonant is not followed by a vowel, then we insert a short vowel after the consonant. For example, consider the word فرح (*farrah*). The first character ف (*f*) (note that Persian is read from right to left) is followed by another consonant ر (*r*). Since ف gets mapped to *f, ph*, we also generate *fa, fe, fi, fo, fu, pha, phe, phi, phu, pho* as candidates for ف.

Table XIII. Persian to English Development Set Results

Persian Dataset	Accuracy (%)					
	Category-wise					
	Indo Arabic	Non-Indo Arabic	In-Vocab	OOV	Overall	MRR
Basic System (N=3)	25.0	24.8	26.6	14.1	24.8	0.135
Enlarge Alphabet (N=3)	23.9	25.6	27.1	14.8	25.3	0.139
GT Disc. + Katz BO (N=6)	28.5	36.5	39.4	9.6	35.1	0.227
CG Disc. + KneserNey (N=7)	20.2	31.4	33.2	7.2	29.5	0.192
PPM-D (N=7)	45.4	50.7	56.8	7.2	**49.7**	**0.359**
MLE (N=7)	30.0	37.7	41.9	3.1	36.3	0.264
Baseline Approaches						
CRF Transliteration	50.9	47.8	-	-	48.3	0.318
SMT Transliteration	81.6	63.9	-	-	**67.0**	**0.502**

Table XIV. Persian to English Test Set Results

Persian Dataset	Accuracy (%)			
	Category-wise			
	Indo Arabic	Non-Indo Arabic	Overall	MRR
Dev				
CRF Transliteration	50.9	47.8	48.3	0.318
SMT Transliteration	81.6	63.9	**67.0**	**0.502**
PPM-D	45.4	50.7	49.7	0.359
Test				
CRF Transliteration	48.8	47.8	48.0	0.325
SMT Transliteration	80.2	64.7	**67.2**	**0.502**
PPM-D	41.8	48.1	47.0	0.343

11.3 Experiments and Results

We repeated the step-by-step procedure given in Table IV. We first experimented with the basic system, then enlarged the alphabet on the English side, and then weighed each English word as per its logarithmic-frequency in the Wikipedia corpus. Since the Persian-to-English character mapping table is not separated by word origin, we could not do the word origin-based steps. We also experimented with various smoothing techniques and found that PPM-D performed the best. The results of our experiments are shown in Tables XIII and XIV. The results follow the trend for Hindi to English transliteration as discussed in Section 5. The performance of our system is comparable to the CRF system while the SMT system does much better. Being a consonantal language, Persian is heavily under determined, and therefore transliterating from it using rule bases alone is harder.

12. CONCLUSIONS

In conclusion, we have shown that for resource-scarce languages, a reasonable transliteration system can be built by judiciously applying statistical techniques to monolingual resources in conjunction with manually created bilingual rule bases. The statistical technique that we focus on is the Character Sequence Modeling (CSM), typically called Language Modeling. It was not properly exploited by the existing systems and rich dividends are obtained by paying proper attention to it. We have also presented an entropy-based explanation for widely varying transliteration accuracy numbers reported in the literature. This analysis explains in general why some transliteration tasks are harder than others.

ACKNOWLEDGMENTS

We thank Dr. Sarvnaz Karimi for providing us the Persian-English dataset and the rule bases. We also thank Dr. Maryam Shojaei Baghini for helping us in understanding issues related to the Persian script. The Indian search-engine company Guruji Inc. provided us with the Hindi Web content, and we are thankful to them. Finally, we would like to thank the anonymous reviewers who helped us improve not just the presentation but also the content of this work.

REFERENCES

AbdulJaleel, N. and Larkey, L. S. 2003. Statistical transliteration for English-Arabic cross language information retrieval. In *Proceedings of the 12th International Conference on Information and Knowledge Management (CIKM'03)*, 139–146.

Al-Onaizan, Y. and Knight, K. 2002. Translating named entities using monolingual and bilingual resources. In *Proceedings of the 40th Annual Meeting of the Association of Computational Linguistics (ACL'02)*, 400–408.

Arbabi, M., Fischthal, S. M., Cheng, V. C., and Bart, E. 1994. Algorithms for Arabic name transliteration. *IBM J. Res. Develop. 38*, 2, 183.

Bilac, S. and Tanaka, H. 2004. A hybrid back-transliteration system for Japanese. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING'04)*, 597.

Cleary, J. G., Ian, and Witten, H. 1984. Data compression using adaptive coding and partial string matching. *IEEE Trans. Comm. 32*, 396–402.

Collier, N., Kumano, A., and Hirakawa, H. 1997. Acquisition of English-Japanese proper nouns from noisy-parallel newswire articles using Katakana matching. In *Proceedings of the Natural Language Pacific Rim Symposium (NLPRS'97)*. 309–314.

Crystal, D. 1992. Graphology: Types of writing systems. In *The Cambridge Encyclopedia of Language* 2nd Ed. Cambridge University Press, 199–205.

Darwish, K., Doermann, D., Jones, R., Oard, D., and Rautiainen, M. 2001. TREC-10 experiments at university of Maryland CLIR and Video. In *Proceedings of the 10th Text Retrieval Conference (TREC'01)*. NIST.

Ekbal, A., Naskar, S. K., and Bandyopadhyay, S. 2006. A modified joint source-channel model for transliteration. In *Proceedings of the COLING/ACL on Main Conference Poster Sessions (COLING'06)*, 191–198.

Frawley, W. J. 1992. Writing and written language: Writing systems. In *International Encyclopedia of Linguistics*, vol. 4, 2nd Ed. Oxford University Press, 383–384.

Ganesh, S., Harsha, S., Pingali, P., and Verma, V. 2008. Statistical transliteration for cross language information retrieval using HMM alignment and CRF. In *Proceedings of the Workshop on CLIA, Addressing the Needs of Multilingual Societies (IJCNLP'08)*.

Gao, W., Wong, K.-F., and Lam, W. 2004. Phoneme-based transliteration of foreign names for OOV problem. In *Proceedings of the 1st International Joint Conference on Natural Language Processing (IJCNLP'04)*. 374–381.

Haizhou, L., Min, Z., and Jian, S. 2004. A joint source-channel model for machine transliteration. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics (ACL'04)*. 159.

Hoang, H., Birch, A., Callison-burch, C., Zens, R., Aachen, R., Constantin, A., Federico, M., Bertoldi, N., Dyer, C., Cowan, B., Shen, W., Moran, C., and Bojar,

O. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, Demonstration Session (ACL'07)*, 177–180.

HUANG, F. 2005. Cluster-specific named entity transliteration. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT'05)*, 435–442.

JURAFSKY, D. AND MARTIN, J. H. 2008. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition* 2nd Ed. Prentice Hall.

KARIMI, S. 2008. Machine transliteration of proper names between English and Persian. Ph.D. Thesis. RMIT University, Australia.

KARIMI, S., TURPIN, A., AND SCHOLER, F. 2007. Corpus effects on the evaluation of automated transliteration systems. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL'07)*, 640–647.

KAWTRAKUL, A. 1998. Backward transliteration for Thai document retrieval. In *Proceedings of the IEEE Asia-Pacific Conference on Circuits and Systems (APCCAS'98)*, 563–566.

KLEMENTIEV, A. AND ROTH, D. 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 (ACL'06)*, 817–824.

KNIGHT, K. AND GRAEHL, J. 1997. BMachine transliteration. In *Proceedings of the 8th Conference on the European Chapter of the Association for Computational Linguistics (EACL'97)*, 128–135.

KUDO, T. 2003. *CRF++: Yet Another CRF Toolkit*. http://crfpp.sourceforge.net.

KUMARAN, A. AND KELLNER, T. 2007. A generic framework for machine transliteration. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'07)*, 721–722.

LI, H., KUMARAN, A., ZHANG, M., AND PERVOUCHINE, V. 2009. Whitepaper of NEWS 2009 machine transliteration shared task. In *Proceedings of the 1st International Joint Conference on Natural Language Processing (IJCNLP'09)*.

LLITJOS, A. F. AND BLACK, A. W. 2001. Knowledge of language origin improves pronunciation accuracy of proper names. In *Proceedings of International Conference on Speech Communication and Technology (EUROSPEECH'01)*, 1919–1922.

MALIK, M. G. A. 2006. Punjabi machine transliteration. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (ACL'06)*, 1137-1144.

MIN, Z., HAIZHOU, L., AND JIAN, S. 2004. Direct orthographical mapping for machine transliteration. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING'04)*, 716.

OCH, F. J. AND NEY, H. 2003. A systematic comparison of various statistical alignment models. *Comput. Linguist. 29*, 1, 19–51.

OH, J.-H. AND CHOI, K.-S. 2002. An English-Korean transliteration model using pronunciation and contextual rules. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING'02)*. 1–7.

OH, J.-H. AND CHOI, K.-S. 2005. An ensemble of grapheme and phoneme for machine transliteration. In *Proceedings of the International Joint Conference on Natural Language Processing (IJCNLP'05)*. 450–461.

OH, J.-H., CHOI, K.-S., AND ISAHARA, H. 2006. A machine transliteration model based on correspondence between graphemes and phonemes. *Trans. Asian Lang. Inform. Process. 5*, 3, 185–208.

PAŞCA, M. 2007. Weakly-supervised discovery of named entities using Web search queries. In *Proceedings of the 16th ACM Conference on Information and Knowledge Management (CIKM'07)*, 683–690.

PADARIYA, N., CHINNAKOTLA, M., NAGESH, A., AND DAMANI, O. P. 2008. Evaluation of Hindi to English, Marathi to English and English to Hindi CLIR at FIRE 2008. In *Working Notes of Forum for Information Retrieval and Evaluation (FIRE'08)*.

ROLLINGS, A. G. 2004. *The Spelling Patterns of English*. LINCOM GmbH.

SAINI, T. S., LEHAL, G. S., AND KALRA, V. S. 2008. Shahmukhi to Gurmukhi transliteration system. In *Companion Volume: Demonstrations*. Organizing Committee for the International Conference on Computer Linguistics (COLING'08) 1, 177–180.

SHERIF, T. AND KONDRAK, G. 2007. Substring-based transliteration. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL'07)*, 944–951.

SHUKLA, S. 2000. *Hindi phonology*. LINCOM GmbH.

SPROAT, R., TAO, T., AND ZHAI, C. 2006. Named entity transliteration with comparable corpora. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (ACL'06)*, 73-80.

STOLCKE, A. 2002. SRILM - An extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP'02)*.

SURANA, H. AND SINGH, A. K. 2008. A more discerning and adaptable multilingual transliteration mechanism for Indian languages. In *Proceedings of the International Joint Conference on Natural Language Processing (IJCNLP'08)*.

TEAHAN, W. J. 1997. Modelling English text. Ph.D. Thesis, The University of Waikato, New Zealand.

UDUPA, R., SARAVANAN, K., KUMARAN, A., AND JAGARLAMUDI, J. 2009. MINT: A method for effective and scalable mining of named entity transliterations from large comparable corpora. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL'09)*, 799–807.

VIRGA, P. AND KHUDANPUR, S. 2003. Transliteration of proper names in cross-lingual information retrieval. In *Proceedings of the Workshop on Multilingual and Mixed-Language Named Entity Recognition (ACL'03)*, 57–64.

WAN, S. AND VERSPOOR, C. M. 1998. Automatic English-Chinese name transliteration for development of multilingual resources. In *Proceedings of the 17th International Conference on Computational Linguistics (COLING'98)*, 1352–1356.

ZELENKO, D. AND AONE, C. 2006. Discriminative methods for transliteration. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing Association for Computational Linguistics (EMNLP'06)*.

Received October 2009; accepted March 2010