TRANSLITERATION BASED APPROACHES TO IMPROVE CODE-SWITCHED SPEECH RECOGNITION PERFORMANCE

Jesse Emond, Bhuvana Ramabhadran, Brian Roark, Pedro Moreno, Min Ma

{emond,bhuv,roark,pedro,minm}@google.com

ABSTRACT

Code-switching is a commonly occurring phenomenon in many multilingual communities, wherein a speaker switches between languages within a single utterance. Conventional Word Error Rate (WER) is not sufficient for measuring the performance of code-mixed languages due to ambiguities in transcription, misspellings and borrowing of words from two different writing systems. These *rendering* errors artificially inflate the WER of an Automated Speech Recognition (ASR) system and complicate its evaluation. Furthermore, these errors make it harder to accurately evaluate *modeling* errors originating from code-switched language and acoustic models. In this work, we propose the use of a new metric, transliteration-optimized Word Error Rate (toWER) that smoothes out many of these irregularities by mapping all text to one writing system and demonstrate a correlation with the amount of code-switching present in a language. We also present a novel approach to acoustic and language modeling for bilingual code-switched Indic languages using the same transliteration approach to normalize the data for three types of language models, namely, a conventional n-gram language model, a maximum entropy based language model and a Long Short Term Memory (LSTM) language model, and a state-of-the-art Connectionist Temporal Classification (CTC) acoustic model. We demonstrate the robustness of the proposed approach on several Indic languages from Google Voice Search traffic with significant gains in ASR performance up to 10% relative over the state-of-the-art baseline.

Index Terms— Code-switching, Transliteration, Multilingual Speech Recognition, Deep Learning

1. INTRODUCTION

Code-switching is common among bilingual speakers of Hindi-English, Bengali-English, Arabic-English and Chinese-English, among many others. When a word from a foreign language (English) becomes part of the vocabulary of a native language (Hindi), the distinction between code-switching, loan words, and creation of new words in the lexicon of the native language is often not very clear, and falls on a continuum [1]. This phenomena renders the transcription of code-switched speech difficult and inconsistent, resulting in the same word being transcribed using different writing systems. These inconsistencies can lead to incorrect count distributions amongst words with similar acoustic and lexical context in both acoustic and language models.

Several approaches have been suggested in the literature for code-switched speech recognition. A straightforward approach is the use of multi-pass speech recognition, wherein regions of code-switching are first identified with language identification methods that use acoustic information only. Subsequently, the speech segments are recognized using the corresponding monolingual acoustic and language models [2, 3]. A framework to run parallel recognizers followed by a second pass merging and rescoring approach was presented in [4]. Such methods not only require multiple passes of processing, but also rely on the accuracy of code-switch detection and language identification. They also fail to capture the context information at the code-switch boundaries. A second approach is the use of multilingual acoustic and language models in a single recognition pass [5, 6]. This approach requires a unified phonetic alphabet and results on the Chinese-English language pair were mixed [7]. While modeling pronunciation variation in lexicons helps in language-pairs such as Ukrainian-Russian, where language identification is problematic [8], other lexicon learning approaches for commonly used homophones have been less successful in improving speech recognition performance [9]. The approach to multilingual language modeling for music information retrieval proposed in [10] utilizes a bilingual dictionary to map the statistical n-grams in one language to the other and reports small improvements in recognition performance on language specific phrase error rates.

In [3], the use of machine translation to generate artificial code-switched text was first explored and integrated with the use of language-identification to identify code switched regions. The use of a weighted finite state transducer (WFST) framework to incorporate a translation model coupled with syntactic constraints to capture contextual information at code switches between Chinese and English was proposed in [11, 12]. Motivated by linguistic studies on the impact of Part-of-Speech (POS) tags on code switching, a recurrent neural network language model with a factorized output layer to predict the next word and its language was proposed in [13, 14]. This work provided significant reductions in perplexity for the Chinese-English language pair but relatively small reductions in Word Error Rate (WER). Reductions in perplexity and marginal improvements in mixed error rate were demonstrated when using combinations of word clusters and POS tags in factored language models [15]. A comparison of several approaches to interpolating individual language models within a single multilingual recognizer to handle code-switching for the German-English language pair is presented in [16] and a multilingual language modeling framework to combine several monolingual into one multilingual LM for code-switching at sentence boundaries is presented in [17]. Both these approaches were able to produce improvements in ASR performance.

In this paper, we propose an alternative strategy based on transliteration for improving ASR performance. WFSTs have been used extensively for speech recognition decoding, where WFSTs representing a context-dependent phone sequence model (C), the pronunciation lexicon (L) and the language model (G) can be composed into a single large transducer ($C \circ L \circ G$, or CLG for short) that maps context-dependent phone sequences to word sequences [18]. In [19], the authors extend the finite-state decoding approach to transliterated keyboards and demonstrate that WFST encoding of transliteration models allows for several optimizations that yield good accuracy. In code-switched languages, the speaker mixes romanized sequences with native language scripts, such as Devanagari in the case of Hindi. Hence, we take a similar approach, where we use transliteration modeling in both the acoustic and language models to capture the code-switches. A series of WFST optimizations achieve the accuracy, latency and memory usage operating points. Transliteration approaches have been used extensively in machine translation [20, 21, 22] and information retrieval [23]. To the best of our knowledge, this is one of the first pieces of work that demonstrates significant gains in ASR performance for Indic languages using a transliteration based approach.

The rest of this paper is organized as follows. Section 2 describes the challenges in speech recognition of Indic languages. In Section 3, we introduce a new metric, transliteration optimized Word Error Rate (**toWER**) and describe our proposed approach and WSFT optimizations. Using Hindi as an example, Section 4 illustrates the importance of evaluating ASR systems by eliminating code-switches and scoring in a *common* transliterated space. Significant performance improvements that can be achieved when incorporating transliteration into the training of acoustic and language models are highlighted. Section 5 validates the generality of the proposed approach in other Indic languages. The paper concludes with a detailed analysis in Section 6.

2. CHALLENGES IN INDIC LANGUAGES

In India, bilingualism is commonplace and code switching between the native language and English occurs frequently. We present the distribution of Latin script seen in two of the corpora used in building language models for Indic languages in Table 1. The corpus containing typed search queries contains far more Latin than the corpus containing transcriptions of spoken queries. We attribute this to transcription conventions that transcribers tried to adhere to, while web-based search queries are not restricted in any manner.

Language	Fraction of Latin in written queries (%)	Fraction of Latin in spoken queries (%)
Hindi	58.36	11.54
Bengali	71.37	16.17
Kannada	81.19	1.76
Gujarati	79.69	9.74
Tamil	71.41	4.14
Malayalam	75.48	-
Urdu	5.14	-
Marathi	81.63	6.49

Table 1. Distribution of Latin script in Indic languages across two different corpora containing written and spoken queries

हेल्लो Hi	Hello Hi
ऐर्टेल् 4 G मोबिल्	Airtel 4G Mobile
सम्सुन्ग् J2 का rate	Samsung J2 ka rate
Satta Matka डाट् काम् साइट्	Satta Matka dot com site
MP3 रिंग् ठोन् नू	MP3 Ring Tone new
Robot 2 फ़िल्म् HD	Robot 2 film HD

Table 2. Examples containing Devanagari and Latin writing systems

Table 1 illustrates the widespread distribution of code-switching. In this paper, we present a detailed analysis of the proposed approach using Hindi, which is one of the languages with a large number of code-switches and training data, while illustrating the generalization of our approach and its impact on other Indic languages as well.

Code-switching is present in multiple writing systems. For example, Hindi uses the Devanagari script, while Urdu uses an Arabic writing system. As most speakers of Hindi are bilingual, code-switching is a part of daily life, the phenomenon routinely occurs in casual conversations, voice search queries and in presentations, leading to what is commonly referred to as *Hinglish*. This type of code-switching can occur within a sentence at a phrase level. A few examples of commonly transcribed spoken utterances are presented in Table 2. The first column illustrates the mixed writing systems used commonly. The second column shows the equivalent text in Latin script for ease of readability and to illustrate the mix of Hindi and English seen in the data. Despite instructions to transcribe Hindi words in Devanagari script and words of English origin in Latin script, numerous inconsistencies can be observed in the resulting transcriptions by bilingual transcribers. Hindi, like other Indian languages, is romanized on social and news media, in user generated text [1], especially with named entity mentions, URLs, numeric entities and acronyms, thereby rendering the transcription of Hindi words in Devanagari even more difficult for the transcribers. These type of inconsistencies directly impact the definition of errors and the metric used for evaluating ASR systems, Word Error Rate (WER). We refer to this as *rendering errors*.

The variability in the usage of the native language (Hindi) and the foreign language (English) makes it challenging to model the context under which code switching occurs. While several methods that use linguistic, prosodic and semantic cues have been proposed to model and capture code-switching phenomena, very few methods have been proposed and been successful in improving the performance of ASR systems. The lack of consistency in transcription and incorrect normalization also impacts the modeling power of language and acoustic models. We refer to this as *modeling errors*. In the next section, we present our unified approach to addressing both modeling and rendering errors.

3. TRANSLITERATION-BASED PROPOSED APPROACH

Transliteration is the process of converting sequences from one writing system to another. While transliteration has been used extensively in improving machine translation and information retrieval performances, to the best of our knowledge, little work has focused on improving speech recognition performance in code-switched languages. For example, in [24], transliteration between Chinese and English was achieved through a joint source channel model, referred to as an ngram-transliteration model, that aligns phonemic units between two writing systems via a bilingual dictionary. Transliteration of Indic languages to Latin script is particularly challenging due to the large combination of consonants, vowels and diacritics that result in a non-unique mapping. It is worth noting that non-standard spellings exist in both scripts, for example, loaner words that have variable possible spellings in Devanagari and Hindi words with variable romanizations.

3.1. General Transliteration Approach

In this paper, we propose the use of transliteration via a weighted finite state transducer as proposed for keyboard decoding in [19]. Transcribers were asked to transcribe spoken utterances in the native writing script (Devanagari, in this case) with exceptions for certain commonly used English words to be written in Latin script. Thus, the context and range of input from the two writing systems was restricted to what was said in the utterance unlike unrestricted text entry via the keyboard. However, given the lack of canonical transliterations between the two writing systems and inconsistencies between transcriptions, a large number of modeling and rendering errors are introduced. The transliteration transducer introduced for keyboard decoding in [19] finds a perfect application for This scenario. The transliteration transducer, T is a composition of three transducers: $I \circ P \circ O$, where I maps input unicode symbols to symbols in a pair language model, P is a bigram pair language model that maps between symbols in the two writing scripts, English and Devanagari, and O maps the pair language model symbols to the target output Devanagari symbols (illustrated in Figure 1). The conditional probability of the transliterated word is obtained by dividing the joint probability from T by the marginalization sum over all input and output sequences. This computation is efficiently implemented by computing the shortest path in T.

3.2. Optimizations

In order to get performance of transliteration at a good operating point with respect to memory, speed and latency considerations for building large-scale language models, we explored several optimizations.

- The transliteration transducer computes the shortest path, and significant speed improvements were obtained by the efficient pruning of the search space. All paths that score below the pruning threshold were discarded. This threshold was determined empirically so as to not impact ASR performance. A prune weight threshold of 5 was determined as a good operating point, particularly as we are only interested in the best path.
- The use of ϵ-transitions to reduce the number of deletions and insertions is important when reducing epsilon cycles in the WFST. We used a parallel implementation of epsilon removal, utilizing eight threads in parallel.
- The operations for epsilon removal caused significant memory explosions, rendering the transliteration process unusable for large scale language models. We addressed this issue via weight-based pruning prior to epsilon removal with no impact on the transliteration performance.
- Given that the bilingual word usage distribution is far from uniform, we observed that several words reappeared very frequently in the training data. We introduced a cache of successful transliterations with a maximum size of 100K elements, thereby reducing transliteration of frequent terms to a table lookup.

The speed-up/memory reduction contributions from the above optimization steps are presented in Table 3. We also explored the use of a unigram language model instead of a bigram model. However, this did not provide any additional speed ups.

The above optimizations reduce the overall training time of a language model trained on 280B words from 165 hours to 12 hours. The transliteration process is only 30% slower than using all of the training data as-is.

Optimization	Speed (msec)
Baseline	123
+ transliteration transducer pruning	109.0
+ parallel epsilon removal	72.6
+ weight based pruning prior to epsilon removal	61.6
+ caching frequent transliterations	25.0

Table 3. Impact of various optimizations on transliteration speed computed over an utterance containing four words on the average

4. DATA, EXPERIMENTS AND RESULTS

4.1. Training Data

All our experiments were conducted on training and test sets that were anonymized and hand-transcribed utterances representative of Google's voice search traffic in Indic languages. The training set is augmented with several copies of the original, artificially corrupted by adding varying degrees of noise and reverberation using a room simulator such that the overall SNR varies between 0 and 20 db. The signal processing pipeline for all languages extracted 80-dimensional log mel-filter bank output features with a standard frame rate of 10 ms. The acoustic models for all languages are LSTMs with 5 layers, with each layer consisting of 768 LSTM cells. The acoustic models were trained in TensorFlow [25] using asynchronous stochastic gradient descent minimizing Connectionist Temporal Classification (CTC) [26] and state-level Minimum Bayesian Risk (sMBR) objective functions [27]. The amount of training data used in our experiments for each of the Indic languages is presented in Table 4. The test data varied between 6K and 10K words. It can be seen that there is a huge variance in available data across these languages. We present detailed analysis on Hindi, as it is one of the languages that has the most code-switching with English and maximum number of training tokens. Our Hindi training data set comprises of approximately 10K hours of training data from 10M utterances. We also validate the proposed approach on the other Indic languages which typically have 10-20% of the data Hindi does.

Language	AM Training Utterances (M)	LM Training tokens
Hindi	10.1M	7.7B
Bengali	2.1M	3.1B
Kannada	0.7M	0.6B
Gujarati	1.4M	0.8B
Tamil	1.3M	2.1B
Malayalam	1.0M	0.9B
Telugu	1.6M	1.2B
Marathi	2.9M	1.7B
Urdu	0.2M	0.3B

Table 4. Training sets for the Indic languages

4.2. Transliterated Scoring of ASR

Table 5 illustrates the significant differences in measured WER after correcting for errors related to the writing systems. The proposed **toWER** metric is computed after transliterating both the reference and hypothesis to one writing system corresponding to the native

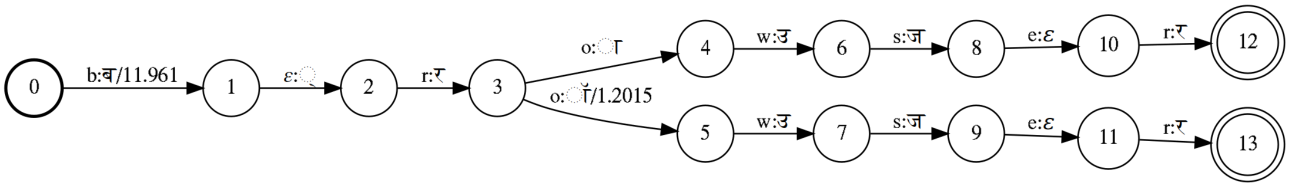

Fig. 1. Illustration of transliteration between Devanagari and Latin scripts for the English word *browser*

locale. It can be seen that there is a correlation between the percentage of Latin script and the proposed metric which serves as a good indication of the extent of code-switching in these languages. In languages such as Malayalam, Telugu, Marathi and Urdu, there is a lesser amount of code-switching than in languages such as Hindi and Bengali and we can see that **toWER** reflects that. Thus, we can see transliteration as a means to correct for errors in the writing system arising from inconsistencies and as a means for separating modeling errors from rendering errors. We hypothesize that transliterated scoring reduces ambiguity introduced by code-switching and smoothes out ambiguities and transcription errors. Therefore, the proposed toWER is a better metric to evaluate any algorithmic improvements.

Language	Baseline WER (%)	toWER (%)	Amount of Latin (%)
Hindi	31.5	20.6	31.9
Bengali	48.1	36.5	31.5
Kannada	36.4	28.1	20.9
Gujarati	38.1	30.3	7.3
Tamil	27.9	25.3	19.0
Malayalam	41.9	38.3	3.1
Telugu	31.2	31.1	3.5
Marathi	25.4	21.0	4.6
Urdu	25.1	23.4	0.4

Table 5. Impact of **toWER** on fixing rendering errors while measuring WER on Voice Search queries on several Indic languages

4.3. Impact of transliteration on Language Modeling

Motivated by the impact of transliteration optimized scoring in Section 4.2, we explored normalization of training data for language models using transliteration. First, we present the results of training transliterated language models (LMs) on Hindi. All the text from the diverse corpora used for building a Hindi language model were first transliterated to eliminate any Latin script. The normalized scripts in Devanagari were subsequently used to train 5-gram LMs for the first-pass and class-based maximum entropy based models for the second pass. Table 6 presents the results obtained when using these models to decode two different test sets comprising of voice search queries and dictation data. In order to compare with various writing systems as inputs to the language model, we define the Devanagari-only data based LM as an LM that was built with all utterances containing Devanagari script only. Any utterance containing bilingual text in Devanagari and Latin scripts was not used in the language model builds. As expected, this resulted in a loss of contextual modeling, lesser data and introduced mismatches between training and test set distributions. Transliterated scoring of the hypotheses produced by this LM fixes mismatches with reference transcriptions (row 2). Retaining data from both writing systems ensures that the contexts from code-switches are preserved but introduces all the challenges discussed in Section 2, including the same word appearing in both, Devanagari and Latin. With all the additional data from Latin included as-is in the LM, the mismatch between the reference and the hypothesis increases even more leading to an artificially inflated conventional WER (row 3). The toWER metric reflects the actual error rate (row 4). It can be seen that retraining LMs with all the data *transliterated* to Devanagari provides a nice gain on the Voice Search and dictation test sets (row 5). Thus, building LMs by transliterating all the training data to Devanagari, thereby introducing consistent text normalization, results in gains of 3 to 8% relative improvements in WER on the two test sets.

Model	Tasks	
	Voice Search (%)	Dictation (%)
Devanagari-only LM (WER)	31.5	14.3
Transliterated Scoring of Devanagari-only LM (**toWER**)	20.6	13.5
LM with Hindi and Latin scripts (WER)	37.0	27.0
LM with Hindi and Latin scripts (**toWER**)	17.7	14.5
Transliterated LM (**toWER**)	17.2	12.7

Table 6. Impact of transliteration on fixing modeling and rendering errors on Voice Search queries and dictation utterances in Hindi with Maximum Entropy based language models

We also explored the impact of transliteration on Long Short Term Memory (LSTM) neural network models. As seen in Table 7, training models with transliterated text provides gains in performance similar to those seen with maximum-entropy based LM for the Voice Search task and less so for the dictation task. While not surprising, it validates our hypothesis that transliteration based normalization for training as well as scoring helps separate *modeling* errors from *rendering* errors and helps with accurate evaluation of the performance of models. For the Voice Search task shown in Tables 6 and 7, one would conclude that the performance of an LSTM LM and a maximum entropy based LM are very similar (32.0 vs 31.5) when using conventional WER, while toWER would suggest that the maximum entropy based LM is much better than the LSTM (20.6 vs 22.3). The significance of such gains can in fact be measured by human raters in a side by side comparison study explained in Section 6.

4.4. Impact of transliteration on Acoustic Modeling

We experimented with transliteration of each utterance in the training of acoustic models (AM) on Hindi. All words in the AM training data written in Latin were first transliterated to Devanagari script and pronunciations were derived in the Hindi phonetic alphabet. After training the model to convergence using the CTC criterion, the

Model	Tasks: Voice Search (%)	Tasks: Dictation (%)
Devanagari-only LM (WER)	32.0	16.0
Transliterated Scoring of Devanagari-only LM (toWER)	22.3	15.3
LM with Hindi and Latin scripts (toWER)	20.7	14.9
Transliterated LM (toWER)	20.2	14.9

Table 7. Impact of transliteration on ASR performance on Voice Search queries and dictation utterances with LSTM LMs

transliterated AM showed small improvements in performance over the model trained with both writing systems (See Table 8). We hypothesize that the wins from sMBR training will be even more significant as the numerator and denominator lattices needed for sMBR training will be consistently rendered in Devanagari script.

Language	Baseline WER (%)	WER using transliterated AM (%)
Hindi	21.9	21.3

Table 8. Impact of our proposed approach on acoustic modeling

5. ASR PERFORMANCE ON SEVERAL INDIC LANGUAGES

Table 9 presents the impact of our proposed approach on several other Indic languages. We observe a significant, consistent gain on all languages except for Malayalam and Tamil. We hypothesize that this can be attributed to the amount of Latin present in the training corpora. For these two languages, it can be seen from Figure 2 that there is very little Latin present in the Voice Search corpus containing spoken queries, while the corpus containing web-based queries contains a lot more Latin. However, the web-based corpus received a very low interpolation weight for this task and therefore had very little impact on the WER. A similar trend is observed with transliterated LMs on the dictation task (See Table 10 with relative reductions in toWER of up to 10%.)

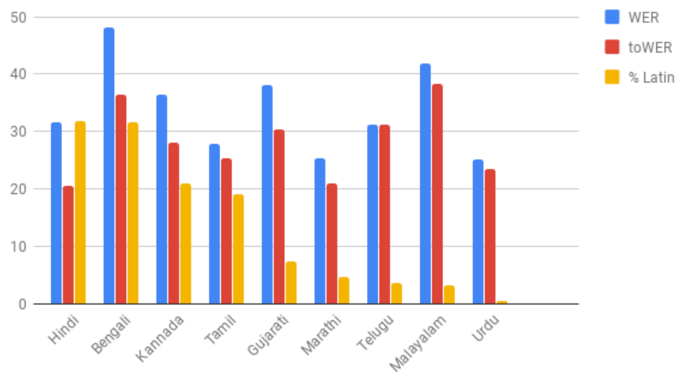

Fig. 2. WER, toWER and correlation with the percentage of code-switching measured as the percentage of Latin in the data

Language	Baseline toWER (%)	Transliterated LMs toWER (%)
Hindi	18.4	17.2
Bengali	36.5	30.5
Kannada	28.1	26.8
Gujarati	30.3	27.1
Telugu	31.1	30.1
Urdu	23.4	22.9
Marathi	21.0	20.8
Tamil	25.3	25.2
Malayalam	38.3	38.5

Table 9. ASR performance on Voice Search queries for several Indic languages

Language	Baseline toWER (%)	Transliterated MaxEnt LMs toWER (%)	Amount of Latin (%)
Hindi	13.5	12.7	4.93
Bengali	18.7	17.6	10.21
Kannada	22.2	23.9	8.64
Gujarati	42.7	38.5	23.09
Tamil	19.2	18.9	1.84
Marathi	22.2	22.0	2.60

Table 10. ASR performance with transliterated LMs on a dictation task for several Indic languages

6. ANALYSIS

In this section, we highlight some of the issues with transliteration with positive and negative examples. Table 11 shows a couple of examples where the conventional WER metric artificially inflates the errors. In the first example, we see the utterance, *Satta Matka* transcribed in Latin script while the ASR system hypothesized in Devanagari and therefore counts as two substitution errors. However, since toWER transliterates to Devanagari before computing the error rate, it correctly produces no errors. A similar scenario can be seen for the word *Discovery* in the second example.

However, not all issues with code-switching can be fixed with transliteration alone. A few such instances are highlighted in Table 12, where the transliteration process introduces errors which did not exist before. In the first example, the utterance in Latin reads as *Tiger zinda hai full movie*. The reference contained the first three words in Latin and the last two in Devanagari. As designed, the ASR hypothesis was in Devanagari. The result of transliterating both the reference and the hypothesis to a common Devanagari writing system, introduced an error *Zinda* vs *Jinda*. Similarly, in the second example, the reference was transliterated to *Jamuna* while the hypothesis produced *Jumna* which is a result of the ambiguity in the transliteration process wherein either forms are acceptable. The third example produces a more classic error. The utterance reads in Latin as *BA first year time table*. Note that in this example, the transcriber was consistent in producing text in Devanagari only. The ASR system hypothesized the utterance correctly but in a combination of writing systems and counted three substitution errors per the WER metric. In the process of transliterating the hypothesis, BA got mapped to *Ba* (pronounced as *bah* in the word 'bar') in Devanagari losing the 'A' at the end of this acronym *BA*. This causes a substitution error with the toWER metric. We would like to point out that despite situations such as those highlighted above, overall, the proposed metric does indeed reflect the performance of the system significantly more accurately than the conventional WER.

As an additional evaluation of the proposed approach, we conducted “side-by-side” (SxS) experiments similar to the ones in [28], in which each anonymized test utterance is automatically transcribed by two ASR systems (Baseline vs. Transliterated LM). If the two hypotheses differ, they are presented to human raters. SxS experiments can accurately measure semantic changes as opposed to minor lexical differences [28]. We conduct SxS experiments also for the scenario where the hypothesis is generated by a single ASR system, but the raters see the raw hypothesis in multiple writing systems as well as in a single native writing system (Devanagari in this example). In Table 13, we present the following results on 500 differing utterances: 1) Change: the percentage of traffic for which the two systems produced different transcripts. 2) Wins/Losses: the ratio of wins to losses in the experimental system vs. the baseline. A p-value less than <5% is considered to be statistically significant.

WER	toWER
Ref: Satta Matka Hyp: सट्टा मट्का	Ref: सट्टा मट्का Hyp: सट्टा मट्का
Ref: डिस्कवरी Hyp: Discovery	Ref: डिस्कवरी Hyp: डिस्कवरी

Table 11. Sample errors *fixed* by toWER

From the table, we see that the human raters give a neutral rating to the transliterated hypothesis when compared to the mixed writing systems based hypothesis. This is not unexpected, as the semantic content of the two systems being compared has not changed. However, toWER smoothes out the rendering errors and offers a better perspective. In a similar vein, the second row compares two LMs, baseline system (row 2 in Table 6 with a toWER of 20.6%) and the system with a transliterated LM (row 4 in Table 6 with a toWER of 17.2%). We see that there are far more wins than losses with the transliterated LM (the experimental system).

	WER	toWER
Ref: Hyp:	Tiger Zinda Hai फ़ुल् मूवी टैगर् जिन्दा है फ़ुल् मूवी	टैगर् ज़िन्दा है फ़ुल् मूवी टैगर् जिन्दा है फ़ुल् मूवी
Ref: Hyp:	Ganga Jumna गं गा जुम्ना	गन्गा जमुना गन्गा जुम्ना
Ref: Hyp:	बीए फर्स्ट् ईयर् टाइम् टेबल् B.A. फर्स्ट् ईयर् time table	बीए फर्स्ट् ईयर् टाइम् टेबल् बा फर्स्ट् ईयर् टाइम् टेबल्

Table 12. Sample errors *introduced* by toWER

Hindi Systems Compared	Traffic Changed (%)	Win/Loss /Neutral	p-Value
Raw vs transliterated hypothesis	12.7	0/3/497	1%-2%
Baseline vs transliterated LM	36.6	57/37/406	>=5%

Table 13. Human Raters judging toWER and transliterated LMs

All Indic languages showed a correlation between the amount of text written in Latin and the gains obtained with transliterated LMs and toWER metric on the Voice Search task. However, we observed a degradation in performance on the dictation task in Kannada. To better understand the source of these errors, we computed the grapheme error rate in the transliterated space for Kannada and compared it with a language such as, Bengali, which showed significant gain with transliterated LMs. Interestingly enough, it can be seen from Table 14 that the number of deletion errors between the baseline and the transliterated LM is much higher in Kannada (increased by 30% relative) than in Bengali (stayed constant). The substitution errors also increased in Kannada by approximately 3% relative, while they decreased by 6.4% relative in Bengali. However, while the grapheme error rate for Bengali reduced from 18.7% to 17.6% with the transliterated LM, it only reduced from 10.27 to 10.23 for Kannada. A closer observation of the hypotheses indicates that many of the errors were either introduced by transliteration when two words are merged into one or were a result of ambiguity from the language where both merged and split forms are considered correct. A combination of these factors ends up degrading toWER. The improvement in grapheme error rate is a good indication that transliterated LMs are still useful. We hypothesize that some of the errors caused by the transliteration process can be corrected by training a model on matched data.

Error Type	LM	Languages	
		Kannada	Bengali
Grapheme Error Rate	Baseline Transliterated LM	10.27 10.23	8.0 7.4
Deletion Errors	Baseline Transliterated LM	4.3 5.6	3.0 3.0
Substitution Errors	Baseline Transliterated LM	14.5 14.9	14.1 13.0

Table 14. Grapheme Error Rate, Deletion and Substitution Errors on the dictation task in Bengali and Kannada. All numbers in %.

7. SUMMARY

We have demonstrated that conventional Word Error Rate (WER) is not sufficient for measuring the performance of code-mixed languages due to ambiguities in transcription, misspellings and borrowing of words from two different writing systems. We demonstrate:

- Accurate measurement of *modeling* errors using the proposed transliteration based toWER metric that smoothes out the *rendering* errors.
- Consistent normalization of training transcripts for both language and acoustic modeling with significant gains of up to 10% relative across several code-switched Indic languages using Google voice search and dictation traffic.

We show that with a simple approach based on transliteration to consistently normalize training data and accurately measuring the robustness and accuracy of the model, significant gains can be obtained.

8. ACKNOWLEDGEMENTS

The authors would like to thank A. Datta, E. Tam, D. Casseiro, A. Waters, J. Apfel and V. Peddinti for several useful suggestions.

9. REFERENCES

[1] Kalika Bali, Jatin Sharma, Monojit Choudhury, and Yogarshi Vyas, "'I am borrowing ya mixing?' An analysis of English-Hindi code mixing in Facebook," in *Proceedings of the First Workshop on Computational Approaches to Code Switching*, 2014, pp. 116–126.

[2] Dau-Cheng Lyu and Ren-Yuan Lyu, "Language identification on code-switching utterances using multiple cues," in *Ninth Annual Conference of the International Speech Communication Association*, 2008.

[3] Ngoc Thang Vu, Dau-Cheng Lyu, Jochen Weiner, Dominic Telaar, Tim Schlippe, Fabian Blaicher, Eng-Siong Chng, Tanja Schultz, and Haizhou Li, "A first speech recognition system for Mandarin-English code-switch conversational speech," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. IEEE, 2012, pp. 4889–4892.

[4] Basem HA Ahmed and Tien-Ping Tan, "Automatic speech recognition of code switching speech using 1-best rescoring," in *International Conference on Asian Language Processing (IALP)*. IEEE, 2012, pp. 137–140.

[5] Dau-Cheng Lyu, Ren-Yuan Lyu, Yuang-chin Chiang, and Chun-Nan Hsu, "Speech recognition on code-switching among the Chinese dialects," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. IEEE, 2006, vol. 1, pp. I–I.

[6] Kiran Bhuvanagirir and Sunil Kumar Kopparapu, "Mixed language speech recognition without explicit identification of language," *American Journal of Signal Processing*, vol. 2, no. 5, pp. 92–97, 2012.

[7] Ching-Feng Yeh and Lin-Shan Lee, "An improved framework for recognizing highly imbalanced bilingual code-switched lectures with cross-language acoustic modeling and frame-level language identification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 7, pp. 1144–1159, 2015.

[8] Tetyana Lyudovyk and Valeriy Pylypenko, "Code-switching speech recognition for closely related languages," in *Spoken Language Technologies for Under-Resourced Languages*, 2014.

[9] Pengcheng Guo, Haihua Xu, Lei Xie, and Eng Siong Chng, "Study of semi-supervised approaches to improving English-Mandarin code-switching speech recognition," *arXiv preprint arXiv:1806.06200*, 2018.

[10] Qingqing Zhang, Jielin Pan, and Yonghong Yan, "Mandarin-English bilingual speech recognition for real world music retrieval," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. IEEE, 2008, pp. 4253–4256.

[11] Ying Li and Pascale Fung, "Code-switch language model with inversion constraints for mixed language speech recognition," *Proceedings of COLING 2012*, pp. 1671–1680, 2012.

[12] LI Ying and Pascale Fung, "Language modeling with functional head constraint for code switching speech recognition," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 907–916.

[13] Heike Adel, Ngoc Thang Vu, Franziska Kraus, Tim Schlippe, Haizhou Li, and Tanja Schultz, "Recurrent neural network language modeling for code switching conversational speech," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. IEEE, 2013, pp. 8411–8415.

[14] Heike Adel, Katrin Kirchhoff, Dominic Telaar, Ngoc Thang Vu, Tim Schlippe, and Tanja Schultz, "Features for factored language models for code-switching speech," in *Spoken Language Technologies for Under-Resourced Languages*, 2014.

[15] Heike Adel, Ngoc Thang Vu, Katrin Kirchhoff, Dominic Telaar, and Tanja Schultz, "Syntactic and semantic features for code-switching factored language models," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 3, pp. 431–440, Mar. 2015.

[16] Zhirong Wang, Umut Topkara, Tanja Schultz, and Alex Waibel, "Towards universal speech recognition," in *Proceedings of the 4th IEEE International Conference on Multimodal Interfaces*. IEEE Computer Society, 2002, p. 247.

[17] Christian Fugen, Sebastian Stuker, Hagen Soltau, Florian Metze, and Tanja Schultz, "Efficient handling of multilingual language models," in *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE, 2003, pp. 441–446.

[18] Mehryar Mohri, Fernando Pereira, and Michael Riley, "Weighted finite-state transducers in speech recognition," *Computer Speech & Language*, vol. 16, no. 1, pp. 69–88, 2002.

[19] Lars Hellsten, Brian Roark, Prasoon Goyal, Cyril Allauzen, Françoise Beaufays, Tom Ouyang, Michael Riley, and David Rybach, "Transliterated mobile keyboard input via weighted finite-state transducers," in *Proceedings of the 13th International Conference on Finite State Methods and Natural Language Processing (FSMNLP 2017)*, 2017, pp. 10–19.

[20] Kevin Knight and Jonathan Graehl, "Machine transliteration," *Computational linguistics*, vol. 24, no. 4, pp. 599–612, 1998.

[21] Nadir Durrani, Hassan Sajjad, Alexander Fraser, and Helmut Schmid, "Hindi-to-Urdu machine translation through transliteration," in *Proceedings of the 48th Annual meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2010, pp. 465–474.

[22] Ahmad Musleh, Nadir Durrani, Irina Temnikova, Preslav Nakov, Stephan Vogel, and Osama Alsaad, "Enabling medical translation for low-resource languages," *arXiv preprint arXiv:1610.02633*, 2016.

[23] Paola Virga and Sanjeev Khudanpur, "Transliteration of proper names in cross-lingual information retrieval," in *Proceedings of the ACL 2003 workshop on Multilingual and mixed-language named entity recognition-Volume 15*. Association for Computational Linguistics, 2003, pp. 57–64.

[24] Haizhou Li, Min Zhang, and Jian Su, "A joint source-channel model for machine transliteration," in *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, Barcelona, Spain, July 2004, pp. 159–166.

[25] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh

Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, Software available from tensorflow.org.

[26] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proc. ICML*. ACM, 2006.

[27] Brian Kingsbury, "Lattice-based optimization of sequence classification criteria for neural-network acoustic modeling," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. IEEE, 2009, pp. 3761–3764.

[28] Fadi Biadsy, Mohammadreza Ghodsi, and Diamantino Caseiro, "Effectively building tera scale MaxEnt language models incorporating non-linguistic signals," *Proc. Interspeech 2017*, pp. 2710–2714, 2017.