Homophone Identification and Merging for Code-switched Speech Recognition

Brij Mohan Lal Srivastava and Sunayana Sitaram

Microsoft Research India

v-bmlals@microsoft.com, sunayana.sitaram@microsoft.com

Abstract

Code-switching or mixing is the use of multiple languages in a single utterance or conversation. Borrowing occurs when a word from a foreign language becomes part of the vocabulary of a language. In multilingual societies, switching/mixing and borrowing are not always clearly distinguishable. Due to this, transcription of code-switched and borrowed words is often not standardized, and leads to the presence of homophones in the training data. In this work, we automatically identify and disambiguate homophones in code-switched data to improve recognition of code-switched speech. We use a WX-based common pronunciation scheme for both languages being mixed and unify the homophones during training, which results in a lower word error rate for systems built using this data. We also extend this framework to propose a metric for code-switched speech recognition that takes into account homophones in both languages while calculating WER, which can help provide a more accurate picture of errors the ASR system makes on code-switched speech.

Index Terms: speech recognition, code-switching, multilinguality, pronunciation, homophones

1. Introduction

Code-switching or mixing is the use of more than one language in a single utterance or conversation, and is common in multilingual communities all over the world. Automatic Speech Recognition (ASR) systems that can handle code-switching need to be trained with appropriate code-switched data that is transcribed in the two (or more) languages being switched. In many cases, the scripts that the two languages use are different, while in other cases, code-switching may occur in language pairs that have similar or the same script, such as English and Spanish. In case the two languages use different scripts, a transcriber may choose to use either language's script while transcribing code-switched speech.

Lexical borrowing occurs when a word from a foreign language becomes part of the vocabulary of a language due to language contact. In some language pairs, the difference between code-switching, mixing and borrowing is often not very clear, and mixing and borrowing can be thought of as a continuum [1]. Due to this, transcription of code-switched speech may not always be standardized, leading to the same word being transcribed using both scripts. This may lead to less data per word for building acoustic models, and more lexical variants for the language model. Ultimately, this may influence the accuracy of ASR systems built with such data.

A related phenomenon exists in monolingual scenarios in the form of homophones in a single language (eg. 'meat' and 'meet'). Typically, a language model is able to disambiguate homophones based on context and pick the correct variant during decoding. Human transcribers are also able to disambiguate such words using context. Additional homophones could occur when languages are mixed, in which case words having the same phonetic realization exist in both languages but have different meanings (eg. 'come' in English and 'कम', meaning 'less', in Hindi). A strong language model built using a large amount of code-switched text should be able to disambiguate such homophones based on context.

In this work, we focus on homophones that are created due to transcription choices and errors. We carry out experiments on conversational Hindi-English code-switched speech and use a common pronunciation scheme to automatically collapse homophones to decrease the Word Error Rate (WER) of the ASR. We also propose a modification to the WER that takes into account potential homophones, which may help give a more accurate picture of the errors made by a code-switched ASR system. Our contributions are as follows:

1. We identify that a large number of homophones are created due to transcription choices and errors in code-switched databases
2. We propose using a common pronunciation scheme to automatically identify and merge potential homophones
3. We propose a modification to the traditional WER metric to take into account homophones, to obtain a better description of code-switched ASR performance

2. Relation to Prior Work

Ali et al. propose alternative word error rates based on multiple crowd-sourced references [2, 3] to evaluate ASR for dialectal speech, where there isn't a gold reference transcription due to non-standard orthography of the language.

Jyothi et al. [4] describe a technique to use transcripts created by non-native speakers for training an ASR system. They model the noise in the transcripts to account for biases in non-native transcribers and estimate the information lost as a function of how many transcriptions exist for an utterance. In an isolated word transcription task for Hindi, they recover 85% of correct transcriptions. Our problem is different from this formulation in that transcribers are assumed to be bilinguals who know the two languages being mixed, and hence the

Table 1: *Hindi-English code switched data*

Data	Utts	# of Spkrs	Hrs	Total Words	En (%)	Unique Words	En (%)
Train	41276	429	46	560893	16.6	18900	40.23
Test	5193	53	5.6	69177	16.5	6000	41.01
Dev	4689	53	5.7	68995	16.05	6051	40.04

noise expected is much less and usually in the form of spelling variants and cross-script transcription.

In this work, we use the WX notation for mapping English and Hindi words to a common pronunciation scheme [5], similar to [6]. The Unitran mapping is a similar scheme that maps all Unicode characters to a phoneme in the X-SAMPA phoneset [7]. The GlobalPhone project proposed by Schultz et al. [8] attempts to unify the phonetic units based on their articulatory similarity shared across 12 languages. Both Unitran and GlobalPhone can be used in place of WX in our proposed framework.

3. Data

The dataset used in this work contains conversational speech recordings spoken in code-switched Hindi and English. Hindi-English bilinguals were given a topic and asked to have a conversation with another bilingual. They were not explicitly asked to code-switch during recording, but around 40% of the data contains at least one English word per utterance. The conversations were transcribed by Hindi-English bilinguals, in which they transcribed Hindi words in the Devanagari script, and English words in Roman script. There was no distinction made between borrowed words and code-switching, which led to some inconsistencies in transcription. Hindi, like other Indian languages, is usually Romanized on social media, in user generated content and in casual communication, which could have contributed to making transcription of Hindi words in Devanagari even more difficult for the transcribers.

The code-switched dataset contains 51158 utterances spoken by 535 different speakers. Hindi words made up 85% of the data, making it the predominant language in this corpus. However, 40% of the total words in the vocabulary were in English. The distribution of Hindi and English words and vocabulary were similar in train, test and dev. A summary of the code switched dataset is shown in Table 1.

4. Homophone merging

Errors and ambiguities in transcription of code-switched speech occur because bilinguals have access to two transcription schemes, which inflate the word error rate of a code-switched ASR system and complicate their evaluation. One approach to solving this problem would be to come up with more standardized schemes for transcription of code-switching, however, this is difficult due to the fuzziness of borrowing vs mixing. Enforcing a standard may be even harder if we rely on relatively untrained transcribers such as crowd-workers to provide transcriptions for audio. Instead, in this work, we focus on approaches that can automatically smoothen out these irregularities by identifying them across the corpus. Due to the nature of our proposed approach, we not only identify homophones, but also spelling variants and errors.

4.1. Common pronunciation scheme

In order to identify words in different languages with similar pronunciation, they must be decoded using a common pronunciation scheme. We choose the WX notation as the pronunciation scheme since Hindi words written in Devanagari can be directly transcribed into WX and using a simple rule-based method the pronunciation of the word can be obtained. To get the pronunciation of a Devanagari word, it is first converted to its WX representation using wxILP[1]. Then, each character is separated and special characters like nukta and anunasika are processed to ignore the character or make it part of the phoneme.

To get the pronunciation for English words written in Roman characters, we train a sequence-to-sequence neural network using CMUDict as training data which takes a sequence of Roman characters as input and produces sequence of CMUDict phonemes. We used Microsoft CNTK [9] to build the sequence-to-sequence recurrent network with Long Short Term Memory (LSTM) cells along with attention on hidden layers. The CMUDict phone sequence obtained by applying this model is then converted to WX using a mapping that we created. If there exists a word-final *schwa*(ə), it is deleted.

4.2. Approach

Result: $Lexicon, Rmap$
EN = {valid English words with frequency};
HI = {valid Hindi words with frequency};
V = {vocab to be merged};
$PMAP \leftarrow$ {pronunciation map};
while v_i *in* V **do**
  $PMAP[pron(v_i)] \leftarrow \{v_i\}$;
end
while *wordlist in PMAP* **do**
  if *len(wordslist)* $==$ *1* **then**
    $Lexicon \leftarrow (w_i, pron(w_i))$;
  else
    Select w_i amongst *wordlist* with highest frequency in EN or HI;
    $Lexicon \leftarrow (w_i, pron(w_i))$;
    $anchor = w_i$;
    $Rmap[anchor] \leftarrow \{wordlist - anchor\}$;
  end
end

Algorithm 1: Homophone merging algorithm which returns the compact lexicon and Rmap. Rmap contains potential suggestions for words that can be replaced (replacee) by an anchor word (replacer). Each group of candidates is then searched for an alpha (anchor word) which must replace others in the group based on its frequency in the corpus.

In our corpus we have around 19k words (tokens), out of which 7k are written in Roman script and 12k are in Devanagari. We found 442 candidate words which could be merged with anchor words. Anchor words (*replacer* in

[1]https://github.com/irshadbhat/litcm/blob/master/litcm/wxILP.py

Type	Replacee	Replacer
Misspellings/ Alternate spelling	benifit	benefit
	x-boyfriend	ex-boyfriend
	compair	compare
	suprise	surprise
	हौकी	हॉकी
	डॉक	डॉक्
	ऑर	और
	थाळी	थाली
Cross-transcription/ Borrowing	टफ	tough
	ब्रेकप	breakup
	स्किन	skin
	सैलरी	salary
	saavan	सावन
	rohu	रोहू
	mochi	मोची
	biriyani	बिरियानी
Misidentified instances	maal	male
	केम	chem
	oks	ox
	uae	you
	lic	लिक
	eg	egg
	umar	अमर
	jonny	जानी
	tt	टीटी
	colours	colour's

Table 2: *Instances of word-pairs identified for homophone merging*

Table 2) are defined as the words which are selected to replace candidate homophones (*replacee* in Table 2). The frequency of these 442 is around 1% that of the words in the entire corpus. It is observed that they are being replaced by more frequent words whose frequency is around 9% that of the words in the entire corpus. We also observe that 288/442 replacements are either English-to-English or Hindi-to-Hindi which indicate misspellings or alternate spellings. 154/442 replacements are either English-to-Hindi or Hindi-to-English which attributes to borrowed words or cross-transcriptions. Table 2 presents some of the instances identified in our corpus using Algorithm 1.

Algorithm 1 accepts a vocabulary of words to be merged in all languages, a list of possibly valid words in all languages, and yields a *Lexicon* for final merged words and a key-value map of words *Rmap*, with key as the word (*anchor*) which will replace all the words in the list mapped as the corresponding value. First, $PMAP$ aggregates all the homophones according to the common pronunciation scheme. Then for each pronunciation, we go through the *wordlist* and disambiguate words as actual homophones or different words. Finally, we select an *anchor* word which will replace all the selected homophone variants and compose *Rmap* using this information.

We trained ASR systems using the Kaldi toolkit [10] with a phoneset of 89 phones, with 39 English and 50 Hindi phones. We used the CMU Pronunciation dictionary [11] as the lexicon for all the English words in the data. To generate the pronunciation for the Hindi words, we used the Festvox Indic front end [12]. We used a word-level trigram language model built on the training transcripts of the code-switched data for all experiments. After merging all such instances (WER comb. in Table 3), we observe around 1.2% relative improvement in the WER as compared to baseline (Baseline comb. in Table 3).

We conducted another experiment to ascertain that homophone merging leads to better ASR performance by reducing the LM perplexity. We trained an RNNLM and an SRILM model using the training transcripts before applying the homophone merging and tested using the test transcriptions. We repeated this after applying the homophone merging and observed a relative improvement of 0.5% in the perplexity of LMs trained over merged data as compared to LMs trained over the original transcriptions.

The errors made by the homophone merging algorithm can be attributed to the following. Firstly, the monolingual vocabularies contain most of the valid words in both languages, so lexical features such as inflections are not modeled and can introduce noise in the pronunciations. Secondly, the mapping between ARPABet and Wx takes into consideration the sound change that occurs when English words are pronounced by Hindi speakers. This can lead to some errors due to the assumption that the mapping is an accurate manifestation of the sound change[13].

In order to overcome these issues the algorithm must take into account the orthographic and phonetic features of each language and build a robust model to disambiguate homophones. In future work, we plan to combine character-level language models with the existing model to make the homophone merging more resilient to the errors mentioned above.

5. Pronunciation-optimized Word Error Rate (poWER)

Conventional word error rate is not sufficient for measuring the performance of code-mixed acoustic models due to cross-transcription, misspellings and borrowing of words. Therefore, we extend the usage of the common pronunciation scheme developed in Section 4 to propose a metric which smooths out these inconsistencies and helps us do a more accurate error analysis of the ASR performance by precluding the inflation of WER caused by such irregularities.

We call this metric pronunciation-optimized word error rate (poWER) and it is defined as the Levenshtein distance between the pronunciation optimized hypothesis (H) and reference (R) sentence, normalized by the number of words in the reference sentence (Equation 1).

$$poWER = D(po(H), po(R))/N \quad (1)$$

$$po(S) = g2p(w_1^S), SIL, ..., SIL, g2p(w_N^S) \quad (2)$$

$$S = w_1^S, w_2^S, ..., w_N^S \quad (3)$$

Here pronunciation optimization $po(.)$ of a sentence S is defined as the grapheme-to-phoneme ($g2p$) conversion of each word in S as per the rules established in section 4 with SIL token inserted between each word to demarcate word boundaries. For example,

Models	Baseline comb.	WER WX	poWER WX	WER comb.	poWER comb.
GMM	52.63	40.21	37.73	52.42	48.72
NNET (DBN Pretrained)	45.42	33.78	31.80	44.88	41.22
TDNN	42.15	31.78	29.97	41.77	38.73
TDNN Discriminative	40.26	29.80	28.03	39.86	36.92

Table 3: *WER and poWER for all systems for two different phonesets. WX phoneset used for acoustic modeling contains 45 phonemes as per the mapping described in section 4. The combined (comb.) phoneset contains 89 phones aggregated from English and Hindi*

Ref: रूम service आपको कैसी लगी
Hyp: room service आपको कैसी लगी

The WER for this pair is 20% whereas the poWER is 0% because "रूम" is a borrowed version of "room" and they can be often cross-transcribed.

We build 4 ASR models using the WX phoneset and the combined phoneset to compare the conventional WER to poWER. WX as a phoneset performs better across all models to the combined phoneset because it minimally models all the pronunciations variants in English and Hindi spoken by a native bilingual speaker. We build a standard GMM-HMM model trained using feature-space maximum likelihood linear regression (fMLLR) as features. The alignments obtained from this model and fMLLR features are used to train a feed-forward neural network (NNET-DBN Pretrained), pre-trained using a stack of Restricted Boltzmann Machines (RBMs) forming a Deep Belief Network (DBN). In addition, we train a *chain* time delay neural network (TDNN) [14] model, and perform discriminative training (TDNN Discriminative) on the TDNN system.

5.1. WER vs poWER comparison

The WER and poWER performance of each of the above systems is listed in Table 3. We observe that the poWER is lower than the WER in all cases and the difference becomes smaller as the acoustic model gets better. This might be because lower phone errors result in a relatively better word prediction accuracy of the model.

Next, we compare WER and poWER for various types of code-switching found in the corpus. We aggregate the WER and poWER of utterances falling in different ranges of the code-mixing index (CMI) [15], defined as linear combination of the fraction of embedded words (in English) in the matrix language (Hindi) and the language alternation points. A higher CMI implies frequent code-switching points and larger phrases of embedded language within the matrix language. It can be compared across varying length utterances since it is normalized by utterance length. From Figure 1 we can see that the difference between WER and poWER is highest when either the CMI is very low or very high. We also see that the WER is highest in these ranges. Utterances with low CMI indicate sporadically embedded words in a matrix language sentence. Such instances are hard to model through a code-switched language model due to the lack of context at switch-points. The predicted words might be among the irregularities mentioned previously; hence the high WER. Such irregularities may be normalized using poWER and a more authentic error rate is obtained reflecting a better transcript. Among the other CMI buckets, poWER performs consistently better than the WER. Both poWER and WER are high at the highest CMI range, which indicates that the utterances have a high number of switch points with almost an equal fraction of both languages, which are highly code-mixed utterances. We observe a steady rise in the WER as the CMI in sentences increases. This follows intuitively from the fact that sentences with frequent code-mixing are more difficult to recognize than those with less frequent code-mixing.

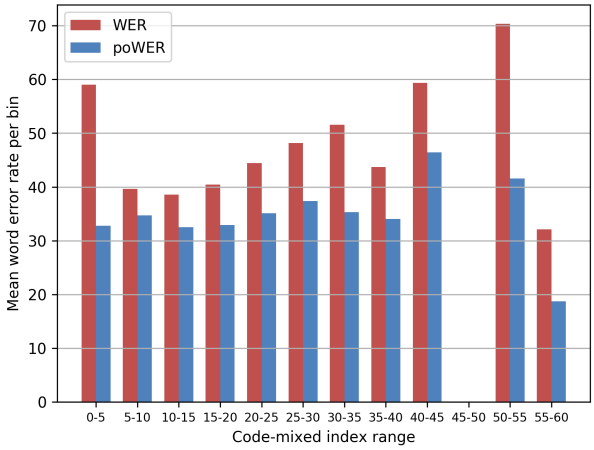

Figure 1: *WER vs poWER comparison aggregated for CMI bins of size 5. The results were computed on the output of our best performing acoustic model (TDNN Discriminative). CMI range 40-45 has no test samples.*

6. Conclusion

In this paper, we introduce the problem of homophone identification and merging for code-switched ASR. We propose a technique to merge homophones in Hindi-English conversational speech. Systems with homophone merging have 1.2% lower relative WER compared to the baseline without merging. Our technique correctly identifies words that have been cross-transcribed, borrowed words, spelling variants and errors. We also propose a modified WER metric called poWER using the technique provided by homophone merging, which takes into account these irregularities to provide a more accurate picture of the performance of the system. We compare WER and poWER across different acoustic models and levels of code-switching in the corpus, and find that poWER can reduce the inflation of WER caused by the complexity of code-switching within utterances, across various types of code-switching.

7. References

[1] J. Sharma, K. Bali, M. Choudhury, and Y. Vyas, ""I am borrowing ya mixing?" An Analysis of English-Hindi Code Mixing in Facebook," *EMNLP 2014*, p. 116, 2014.

[2] A. Ali, W. Magdy, and S. Renals, "Multi-Reference Evaluation for Dialectal Speech Recognition System: A Study for Egyptian ASR," in *Proceedings of the Second Workshop on Arabic Natural Language Processing*, 2015, pp. 118–126.

[3] A. Ali, P. Nakov, P. Bell, and S. Renals, "WERd: Using social text spelling variants for evaluating dialectal speech recognition," *arXiv preprint arXiv:1709.07484*, 2017.

[4] P. Jyothi and M. Hasegawa-Johnson, "Acquiring Speech Transcriptions Using Mismatched Crowdsourcing." in *AAAI*, 2015, pp. 1263–1269.

[5] R. Gupta, P. Goyal, and S. Diwakar, "Transliteration among Indian Languages using WX Notation." in *KONVENS*, 2010, pp. 147–150.

[6] A. Pandey, B. M. L. Srivastava, and S. V. Gangashetty, "Adapting monolingual resources for code-mixed hindi-english speech recognition," in *Asian Language Processing (IALP), 2017 International Conference on*. IEEE, 2017, pp. 218–221.

[7] T. Qian, K. Hollingshead, S.-y. Yoon, K.-y. Kim, and R. Sproat, "A Python Toolkit for Universal Transliteration." in *LREC, Malta*, 2010.

[8] T. Schultz and A. Waibel, "Language-independent and language-adaptive acoustic modeling for speech recognition," *Speech Communication*, vol. 35, no. 1-2, pp. 31–51, 2001.

[9] F. Seide and A. Agarwal, "CNTK: Microsoft's Open-Source Deep-Learning Toolkit," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 2135–2135.

[10] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, "The Kaldi speech recognition toolkit," in *IEEE 2011 workshop on automatic speech recognition and understanding*, no. EPFL-CONF-192584. IEEE Signal Processing Society, 2011.

[11] R. L. Weide, "The CMU pronouncing dictionary," *URL: http://www. speech. cs. cmu. edu/cgibin/cmudict*, 1998.

[12] "The Festvox indic frontend for grapheme to phoneme conversion, author=Parlikar, Alok and Sitaram, Sunayana and Wilkinson, Andrew and Black, Alan W," in *WILDRE: Workshop on Indian Langauge Data, Resources and Evaluation, 2016*.

[13] J. G. Heath, "Language contact and language change," *Annual review of anthropology*, vol. 13, no. 1, pp. 367–384, 1984.

[14] V. Peddinti, D. Povey, and S. Khudanpur, "A Time Delay Neural Network Architecture for Efficient Modeling of Long Temporal Contexts," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[15] B. Gambäck and A. Das, "Comparing the Level of Code-Switching in Corpora." in *LREC*, 2016.