Romanization of Hindi words with natural English pronunciations

Naveen Kumar

Abstract

Romanization is very popular for Indian languages because of the ease of typing. However most of the existing standards for romanization of Hindi or other languages of Indian origin can hardly be called transliteration. They are usually either just adhoc varying from person to person or use simple graphemic rules that fail to get at the actual pronunciation. Transliteration to English is infact not trivial, because of the lack of any strict phonetic structure in English. English speakers typically learn to pronounce by assocation with other "similar" words. In this work I try to exploit this to make the romanization better mimic the original pronunciation of a Hindi Word. The hypothesised *pronunciation by association* model is learnt using Expectation Maximization on an english pronunciation dictionary.

1 Introduction

Though the roots of romanization can be traced back to the colonization era, there has been a radical shift in base since then. Romanization nowadays is largely used as an alternative script for foreign languages. This for example, is very common with Indian languages like Hindi because of its ease as an input method. As a result most of the existing romanization schemes are aimed at Indian native speakers. Most of these schemes hence end up being either graphemic, or are effected by adhoc media standards. With the spread of internet based media like chatting and blogging these have now become standards romanizations for these words. This version of romanized Hindi, often called *Romanagari* (after *Devanagari* the Hindi script) has now become the standard for names of Indian places, movies, and even for Hindi loan words in English. Infact it is commonly used by foreign learners of Hindi as an alternative to the *Devanagari* script which can be intimidating at first. Although these romanizations often carry enough information to be understood by a native Hindi speaker, they do not faithfully map the pronunciation for English/ foreign speakers who do not have any knowledge of Hindi alphabets.

Romanization is infact just a specific type of transliteration. In the broader task of transliteration the objective is to make the word in the target language match the source pronunciation as closely as possible. Hence, we worry about pronunciation rules of the target language and mapping between the intermediate pronunciation layers (Figure 1). For example if a word was being transliterated from English to Hindi, the ideal transliteration would be expected to sound natural when pronounced as a Hindi word. This is where *Romanagari* differs from the standard notion of transliteration. To standardise

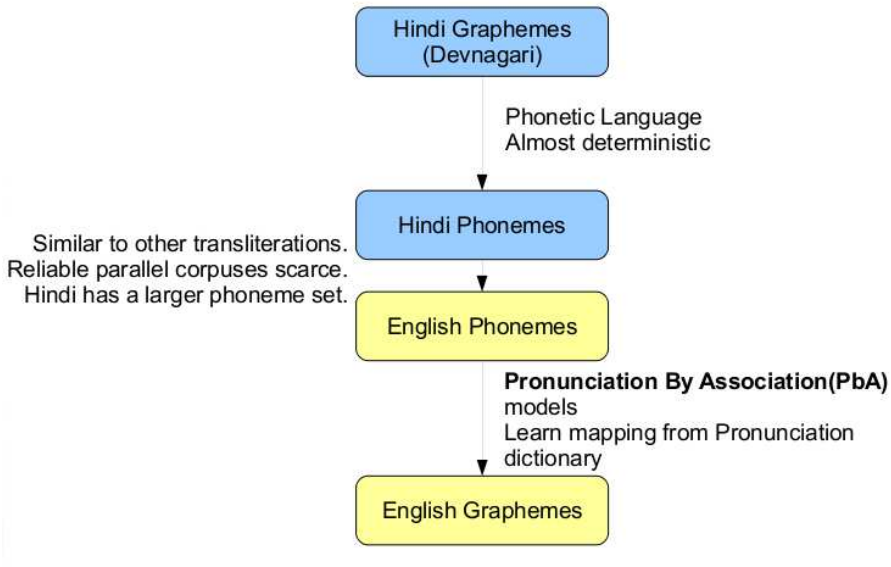

Figure 1: Romanization Scheme from Hindi to English

romanizations, would be to make them dependent only on the rules of English and independent of the source language. This could possibly benefit any non Hindi speaker with the ability to pronounce English words.

2 Basic Romanization Scheme

As seen in Figure 1 the general transliteration scheme comprises of the following parts. They are discussed specifically in the context of romanization below:

2.1 Hindi Graphemes-Hindi Phonemes

Hindi being a strictly phonetic language (pronounced exactly as it is written), pronunciation rules are pretty straightforward. Even then there are a few context based rules which deal with the placement of *Schwa* or some nasal consonants. A list of these rules is given in [1] and was implemented for the purpose of this project. This list of rules although not exhaustive provides transcriptions of sufficient accuracy for our purpose.

2.2 Phoneme-Phoneme mapping

Mapping phonemes from one language to another is a typical task in transliteration schemes. This can be learnt from parallel transcriptions of the data with some language specific knowledge [2]. Typical challenges faced here are many to one or missing mappings, because of a unique phoneme set in one of the language. Hindi typically has a richer phoneme set, compared to English, which makes it harder to learn the mappings to English phonemes. A data-driven approach to learn this mapping is also hard, because of the scarcity of reliable romanization data, as stated above. Additionally, such a method would not be suitable for a Hindi to English transliteration task, simply because some

of the Hindi phonemes would never occur in the parallel corpus because of the absence of any English phonemes to map to.

This problem hence requires a deeper and more informed examination. In this work I try to circumvent this problem, by proposing the use of a common phoneme layer. For convenience I choose the English 39 phoneme set used in CMUdict [3]. The Hindi graphemes are mapped directly into the English phoneme set, as suggested in [4], where the authors use this to build Hindi pronunciation dictionaries for Automatic Speech Recognition using English acoustic models. This direct transcription scheme is implemented in `transcribeHindi.py`.

2.3 English Phonemes-English Graphemes

This layer is unique to a romanization task. Romanization viewed as a transliteration problem means that ideally we would expect the English words output by the system to make intuitive sense to an English speaker, in terms of the pronunciation. One should be able to read the word like any ordinary English word, without any background knowledge of Hindi phonetics.

This would require some sort of modeling of the pronunciation rules in English. English which contains loan words from a multitude of languages, typically has very context dependent pronunciations. This explains how speakers of English usually learn pronunciations by associating with similar words. For example, if you were taught repeatedly that words ending in `-tion` are pronounced `SH AH N` then you are quite likely to pronounce a new word that you seen ending in `-tion` similarly. Other examples maybe words like `(caught, draught), (wine, dine)` etc. I hypothesise that speakers of English would use these Pronunciation by Association rules for any word written in English. This motivates us to model this mapping for this task.

Fortunately we can use data driven approaches to learn this mapping because of the abundance of parallel corpora of English words and their pronunciation. It is important to be noted here that the choice of the pronunciation dictionary can play a crucial role, since it represents the vocabulary of the English speaker. For this work I use the CMUdict American English pronunciation dictionary [3], which lists out pronunciations for over 120000 words, using a set of 39 english phonemes. The mapping is learnt using Expectation Maximization (EM) over the latent alignments. This phoneme to grapheme mapping is represented as a WFST. Specific challenges in using EM for this task have been described in Section 4. Additionaly a trigram character LM is used in the form of a WFSA to smooth the output results as proposed in [5]. **CARMEL** was used to perform the WFST compositions.

3 Hindi Dataset

The Hindi data for this task was taken from a parallel transliteration corpora provided for an open source project called *Wordmint* [6]. It is common for users to write romanized Hindi songs lyrics on websites. This dataset provides song lyrics for 100 Hindi songs written in *Devnagari* and romanized Hindi (*Romanagari*. For the task the corpus was manually checked for obvious spelling errors and a parallel list of about 3000 Hindi words with their romanization was created. Different users often tend to romanize the same

hindi word differently. Hence the corpus contains multiple Romananagari alternatives corresponding to each word. In this work the most frequent romanization for each word in the database is used as the baseline against which our approach is compared.

4 Pronunciation by Association

CMUdict provides a list of English words and their corresponding pronunciations. However the alignments between the graphemes and phonemes are not known. EM is used for this purpose to marginalise over the unknown alignments, and iteratively find a set of mapping rules, that maximise the corpus probability.

Specifically we are interested in learning probabilities of the type $P(\bar{g}|\bar{\phi})$ where $\bar{\phi}$ is a sequence of phonemes and $\bar{g}$ is a sequence of graphemes as shown in Figure 2. At each iteration for the E-step we compute the total probability for each word pronunciaiton pair through all possible alignments, using the forward backward algorithm. This is implemented in `em3.py`.

$$
\begin{aligned}
P(G, \Phi) &= \sum_{z \in align} P(G, \Phi, z) \\
&= \sum_{z \in align} \prod_{z_i | i=1}^{t} P(\bar{g}_i, \bar{\phi}_i, z_i)
\end{aligned}
$$

In terms of the generative story of the model this means that both the words and pronunciation are split into t parts, where each part is represented as a joint probability in our model. The above word pronunciation pair probabibilities can then be used in the M-step which estimates these joint probabilities in a maximum likelihood fashion.

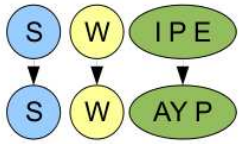

Figure 2: Modeling english pronunciations requires mapping multiple graphemes to multiple phonemes.

An additional challenge here is to decide the maximum permissible lengths for $\bar{g}$ (K) and $\bar{\phi}$ (L). Large K and L values can lead to an exponentially large number of alignments. While problem specific information is generally used to set these constraints, with English, it is in general hard to estimate how much context might be enough. The typical solution used for this is brute force or searching for alignments upto the full length of the word. This is not feasible in our case. Other than being highly inefficient, EM typically favors larger K and L (Figure 3), and shorter derivations. In addition to overfitting, this can reduce the flexibility of the model in representing hindi words.

Since there is a tradeoff involved, EM is tested for different parameters K and L. The number and nature of probability rules generated is used as a rough criteria for selection. Additionally for each parameter set the performance is also compared on a held out set

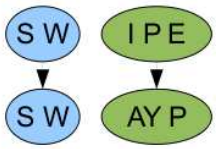

Figure 3: EM prefers shorter derivations : *This derivation might be preferred to the one show earlier, because lesser parts would lead to a higher probability for this alignment.*

of hindi words transcribed into english phonemes. A possible alternative scheme could be to weight different probability rules using a prior based on the length of the sequence. This would be similar to a bayesian version of the EM, where the M-step performs a MAP estimation instead of a ML smoothing. A dirichlet prior was used which prefers smaller phoneme or grapheme sequence over larger ones. Specifically the M step was modified to

$$P(\bar{g}_i, \bar{\phi}_i) = \frac{C(\bar{g}_i, \bar{\phi}_i) + P_i}{\sum_i C(\bar{g}_i, \bar{\phi}_i) + P_i}$$
$$P_i = \lambda(K + L - len(\bar{g}_i) - len(\bar{\phi}_i)), \qquad \lambda \in (0, 1)$$

An extensive evaluation of this scheme however could not be done because of the need to tune λ on a dev set, which is very time consuming for our task. Hence the results were obtained for the ordinary ML EM using the empricially determined values $K = 3$, $L = 3$.

The mapping learnt using this is converted into a WFST by `probs2wfst.py` and stored in `espell-epron.wfst`. The character trigram language model is trained on the words in CMUdict and stored as `espell.wfsa`.

5 Evaluation and Results

As mentioned earlier, this problem by definition suffers from lack of golden romanizations to compare against. This makes automatic evaluations impossible for this task. Even if we had access to oracle romanization, the task of judging transliteration quality is essentially a perceptual task, and it would be unfair to use string distance metrics to compare them. Therefore for this project I resort to human subjective evaluations. This however greatly limits the number of words that can be evaluated. I randomly choose a subset of 100 words from the corpus for this evaluation.

Specifically a native english and hindi speaker are used for the experiment. For each Hindi word, I output the top two hypotheses obtained from the noisy channel model, after composition with the phoneme-grapheme WFST and the character LM. To this list we add the baseline romanization for each word, which was extracted from the corpus as described earlier. This gives a list (`eval/english.txt`) of three candidate romanizations for each Hindi word. The english speaker was given this list and was instructed to pronounce each word in a spontaneous fashion. The hindi native speaker, who was given a list with the hindi words (`eval/hindi.txt`), was asked to rank each pronunciation (`eval/hindiEvals.txt`) of the native english speaker. Additionally the order of the words was randomised to remove any bias on the part of the

hindi speaker(`eval/randomOrder.txt`). From this experiment the following metrics of performance were computed against the baseline.

Criteria againt baseline	Percentage
Best output comparable or better	71
One of the 2-best outputs comparable or better	81
Best output strictly better	54

Statistical tests suggest that the results are better than the baseline at the 1% significance level[1].

6 Conclusion

From this small experiment, we see that pronunciation motivated schemes for romanization, can help making pronunciation more intuitive for non native speakers. One thing to keep in mind though is the importance of choice of the pronunciation dictionary. Depending on the target reader, this should be changed to incorporate suitable vocabulary.

Another challenge at this stage is the absence of any suitable language models for smoothing the output. In the case of noisy channel model problems, language models are well known to boost the performance radically. Future work might focus on using language models from clean hindi romanizations. Additionally it might be interesting to look into other deterministic schemes motivated by linguistic rules like syllable boundaries for this task.

References

[1] M. Choudhury, "Rule-based grapheme to phoneme mapping for hindi speech synthesis," in *90th Indian Science Congress of the International Speech Communication Association (ISCA)*. Citeseer, 2003.

[2] K. Knight and J. Graehl, "Machine transliteration," *Computational Linguistics*, vol. 24, no. 4, pp. 599–612, 1998.

[3] J. Kominek and A.W. Black, "The cmu arctic speech databases," in *Fifth ISCA Workshop on Speech Synthesis*, 2004.

[4] M. Kumar, N. Rajput, and A. Verma, "A large-vocabulary continuous speech recognition system for hindi," *IBM journal of research and development*, vol. 48, no. 5.6, pp. 703–715, 2004.

[5] R. Bhagat and E. Hovy, "Phonetic models for generating spelling variants," in *Proceedings International Joint Conference of Artificial Intelligence*, 2007.

[6] "Transliteration corpus for wordmint, *http://www.wordmint.org/2009/05/transliteration-corpus-for-wordmint*," .

[1] A one sided hypothesis test was done for a binomial distribution, Null Hypothesis p=$\frac{1}{2}$