Invited Talk:

Quillpad Multilingual Predictive Transliteration System

Ram Prakash H
Tachyon Technologies P Ltd, Bangalore, India
`ramprakash@tachyon.in`

ABSTRACT

Transliteration has been one of the common methods for multilingual text input. Many earlier methods employed transliteration schemes for defining one to one mapping of input character combinations to output character combinations. Though such well defined mappings made it easier to write a transliteration program, the end user was burdened with learning the mappings. Further, though transliteration schemes try to map the character combinations phonetically, it is unavoidable to introduce non intuitive combinations into the scheme. An alternative is to use predictive transliteration, where user can input a word, by intuitively combining the input alphabet phonetically and the predictive transliteration system correctly converts it to the target language. This paper presents the challenges that must be addressed to build such a system, and describes how Quillpad can be trained for performing predictive transliteration between any two scripts.

KEYWORDS : transliteration, predictive text input, multilingual text input, Quillpad, decision trees

1 Introduction

Predictive transliteration has proven to be an effective method for text input of Indian languages. This paper discusses general challenges involved in building a practical predictive transliteration system. Further, it describes the key aspects of Quillpad multilingual transliteration system that pioneered such an input method for Indian languages. Though the techniques employed in developing Quillpad system are illustrated in the context of English to Hindi transliteration, the system is generic and can be directly applied to train a system for transliterating between any two alphabet-based languages. Quillpad itself has been successfully trained for phonetic transliteration between Roman input and 16 Indian languages (including Urdu) and Arabic.

The content of this paper will be organised as follows. Section 2 will talk about specific challenges to be addressed by a phonetic transliteration system. Section 3 will discuss the Quillpad system and describe the approaches behind its key modules. Section 4 will briefly discuss some of the issues that have not been addressed by current Quillpad system and can provide some topics for future work.

This document assumes the input language as Roman alphabet and output language as Hindi. Other language examples are used wherever it is appropriate to highlight challenges that may be specific to those languages.

2 Predictive Transliteration Challenges

Predictive transliteration systems should allow users to type in phonetically, by intuitively composing letters from an input alphabet. For the sake of illustration, consider transliterating from Roman alphabet to Hindi. This is an important use case as almost all computers shipped in India come with a Roman keyboard. Predictive transliteration enables users to type in their own language. The following sections highlight the challenges involved in building an effective transliteration solution.

2.1 Phonetic Mapping Ambiguities

Rigid transliteration schemes use upper case and lower case letters to represent similar but different sounds, which generally makes the typing interface cumbersome for the end users. For better usability, the input should be assumed case insensitive. Given this design decision, there are 26 letters in Roman alphabet and 50+ different letters in Hindi. This naturally leads to ambiguous mappings. For example, letter ‘d’ will be used to represent both ‘द’ (<u>th</u>is) , ‘ड‘(<u>d</u>i<u>d</u>) and ‘ड़’. Another example is the letter ‘n’. ‘n’ can be used by the user to write ‘ञ’ , ‘ण’ (used in Marathi), ‘न’(<u>n</u>ot) and nasal markers like ‘◌ं’ (bindu) and ‘◌ँ’ (chandrabindu). Phonetic transliteration system should correctly convert the input character to the corresponding Hindi character depending on the word that is being written. Other cases include different Roman letters mapping to a same Hindi character, cluster of Roman characters mapping to a single Hindi character etc.

2.2 Loosely Phonetic Nature

Though Indian languages are generally phonetic, there are cases where one cannot assume strict phonetic nature. This problem is more evident in Hindi. For example, the word ‘बचपन’ is pronounced as ‘bachpan’(IPA: ‘bətʃpən’) and not as ‘bach<u>a</u>pan<u>a</u>’ which would be the pronunciation if the language were to be strictly phonetic. In Hindi, the final consonant is pronounced without the inherent schwa, though the consonant is written with the implicit vowel sound ‘ə’. South Indian languages like Kannada, Tamil, Telugu do not follow this rule. Though the rule for handling the final consonant is easy to specify, the challenge is in handling the case where vowel sound ‘ə’ is skipped in the middle of the word. In the above example, though the word is written as ‘ba cha pa na’ in Hindi, user would type the word as ‘bachpan’, as per the pronunciation of the word. In general, when two consonants come together, they should be combined into a conjunct ligature. However, in this case though ‘ch’ and ‘p’ are adjacent without any explicit vowel in the input, they should not be combined to form the conjunct ‘च्प’. If the Roman input, ‘bachpan’, is interpreted to be strictly phonetic, the output should have been ‘बच्पन’ which is not what the user intends to write. This condition can’t be defined as the application of this rule is fairly complex and can appear at multiple locations in a given word.

Other extreme cases of this loose phonetic nature can be observed in Arabic and Urdu, where in the output language the vowel sounds are optional in many cases. However, the vowels are explicitly written in the input word as the pronunciation of the word would have the vowel sounds.

2.3 Multiple Input Variants

As a result of issues discussed in 2.1 and 2.2, and other scenarios discussed below, different users can type a given word with multiple variants of input spellings. This problem is compounded by local conventions for writing certain letters or syllables, influence of the native language of the input alphabet and the pronunciation differences due to user's own native tongue influence. Some of these issues are briefly discussed here.

2.3.1 Phonetic Variants

These are generally due to the issues discussed in 2.1 and 2.2. For illustration, consider a Hindi word राष्ट्रपति (IPA: rɑːʃtrpəʈ̪ɪ). Users can write any of the following Roman inputs for writing this word. 'rashtrapati', 'rashtrapathi', 'raashtrapathy', 'raashtrpati' etc. All these inputs should yield the correct Hindi word राष्ट्रपति.

2.3.2 Native Language Influence

The native language of the user plays a significant role in determining the phonetic spelling they come up with for writing a word in their language. For example, a Tamil speaking user wanting to write in Tamil is likely to type the input as 'pirakash' for writing 'பிரகாஷ்' (IPA: prəkɑːʃ). Most other users would type 'prakash' as input. This is because of how the conjuncts are written in native Tamil words. There are many such local language influences that an effective transliteration system should allow to be incorporated into.

2.3.3 Foreign Language Influence

Users fluent in English may use some spelling conventions that are common in native English words. Such combinations are not phonetic, but many of those combinations are commonly used. For example the Hindi word हम(IPA: həm). Though the phonetic input for this word is 'ham', uses familiar with English might type in 'hum' as input. However, if 'hum' is interpreted phonetically, it would result in 'हुम', which is not the intended output. This is a simple example, but there are many more cases where native English spellings can influence user's input.

2.3.4 Conventional Spellings

There are local conventions for representing certain sounds in regional languages. These conventions have been traditionally set in because of multiple reasons. One, when the local language character does not have any character in English that closely represents its sound. For example 'zh' in Tamil and Malayalam is used to represent sounds that are actually not pronounced as 'z'(IPA). Two, historically certain names are written with spellings that do not faithfully represent its phonetic form in the Indian languages. Consider the name 'Krishnaiah', which is actually represented as 'Krishnayya' in Indian languages. One can find many such examples.

2.4 Multiple Output Words

While previous sections discussed about the variants in inputs, it is also possible for a given input word to have multiple valid outputs. Transliteration system should be able to suggest all those possibilities.

2.5 Transliteration of Foreign Words

It is quite common for the users to type native English words in the middle of their Hindi sentences. As the input alphabet they use is same as the alphabet used for English language, they would naturally enter native English spelling for the word. Most of these spellings firstly, are not phonetic, secondly, do not really represent any native Hindi words. A good transliteration system should allow the users to type native English spellings, and if detected as an English word, should convert it into regional language using English pronunciation of the word. This feature significantly improves user experience as it contributes to fluency in writing in local languages. However, it should be kept in mind that native English pronunciations when represented in Indian language scripts, do not capture the local pronunciation of those words. So, it is important for the transliteration system to convert the native English words into a pronunciation that is acceptable locally, and further use the language specific conventions for representing native English phones.

3 Quillpad System Description

Quillpad transliteration system effectively addresses most of the above mentioned challenges. The solution has been live on www.quillpad.in since 2006. This section briefly discusses the overall approach behind the Quillpad system.

3.1 Overview of the Approach

Quillpad has been designed by modelling the core transliteration task as a multi-class classification problem. It can be trained to transliterate between any two alphabet-based languages. The rough mappings between the alphabet code points of input and output languages is definable using a simple language definition file. Once that is done, rest of the pipeline is automatic. The learning and prediction modules do not assume anything specific to the language pair and almost all the language specific issues mentioned above are completely definable in the external language definition file. Once the definition file is ready, which normally takes 8-10 hours to perfect, the predictive transliteration rules are learned given just the target language corpus, without requiring any sort of parallel transliteration corpus. The prediction itself is just a decision tree traversal for each of the input characters and the words are combined and ordered using a language model. Brief descriptions of these steps are given below.

3.2 Alphabet Mapping Definition

Language definition file that defines the mapping between input and output alphabet character is a simple, regular expression based set of rules to specify what English characters users are likely to use for a given Hindi character. By allowing regular expression models, these options can be controlled and many language specific conventions and transliteration rules can be easily captured. The rules themselves are defined per target character, independent of any word the character would appear in. Thus, defining them does not require deep linguistic expertise and the mappings can be one to many, many to one, and many to many. It must be noted that language definition does not define any rule for actually determining which output character a given input character would be converted to. These are defined per character, and theoretically can be just a dump of all possible input alphabet combinations that a user may enter to type the given character. However, doing so increases the artificial training data exponentially. This point will be further clarified in the next section. Modelling the possible input characters for a given Hindi character, in a context-specific manner, will give the model expressive power to incorporate language specific nuances.

3.3 Training Data Generation

One of the important aspects of Quillpad transliteration system is that it doesn’t require a manually prepared transliteration parallel corpus. The alphabet mapping rules are used to generate several possible ways in which the user may type a Hindi word from the corpus. For each Hindi character, the language definition file defines possible English characters the user may use to type it. Such candidates for each Hindi character are combined together to produce multiple possible inputs for a given word. The context specific rules in the language definition model allows one to define rules that would select different candidate English character combinations for a given Hindi character depending on, say, whether that character appears at the beginning of the word, middle of the word, is followed by a particular consonant etc. This controls both the quality and the number of options generated for each corpus word. This significantly determines the quality. Using the English input options generated for every corpus word, one to one correspondence parallel training data is generated.

3.4 Training

Once the training data is generated, a different classifier is trained for each English character. Quillpad uses a decision tree based learning algorithm. The features can be as simple as checking if at a given relative position there is a particular English character. However, it significantly improves the generalisation accuracy of the system if higher level and richer features are used. Some of the higher level features like, if the previous character is a ‘consonant’, or ‘nasal’ will prevent the decision tree from splitting on every character as it can be split by checking a condition on a class of characters. For every Hindi character, arbitrary number of these special labels can be assigned in the language definition file. The learning algorithm automatically makes use of those labels to generate these higher level features. Another important aspect for designing the features is to make it invariant to the exact position of the feature in the input string. Quillpad learning algorithm generates such position invariant features, which are independent of any language.

The actual training for a corpus with 5,00,000 unique Hindi words takes about 30-40 minutes. Once the training is complete, it can be used for prediction.

3.5 Prediction

Prediction module takes the decision tree model for each of the input alphabet characters and applies them on every input character independently. The resulting class assignments for each of these characters are combined together to form candidate output words.

3.6 Pre-Processing

Some of the simple issues mentioned in section 2 can be easily handled by pre-processing the input string before passing the input to the prediction engine. Quillpad system makes provision for language specific pre-processing rules. Though this can be addressed by defining appropriate rules in the language definition file, in practice it is more efficient to deal with some of these cases at a pre-processing level.

3.7 Post-Processing

The generation of candidate words as described in 3.5 is restricted by using beam search. Beam search uses the log probabilities returned by the decision trees for determining the words to be trimmed off the list. By incorporating word frequency score during training, the decision tree also serves as a character level language model. This ensures that log probabilities from the trees can be used reliably in the beam search. The candidate lists are further ordered by using a word level language model to present multiple possible output options to the user.

3.8 Dealing with Foreign Words

Quillpad effectively deals with the challenge in handling native English words in the middle of Hindi text. A simple dictionary lookup is used to check if a given word is a valid English word. However, if the given input is also a valid representation of any word in the corpus, it is not considered as a foreign word. Since the phonetics of English and Indian languages are different, such a simple method works very well. Once the input is determined as an English word, a different prediction model, similar to the one used for Hindi prediction, is used to convert the input into its Hindi representation. A different set of decision trees are used for this purpose as mixing native English pronunciation rules significantly differ from those of Hindi prediction. Training them as separate models helps improve both prediction and generalisation accuracy.

4 Conclusion & Future Scope

Quillpad, multi-lingual predictive transliteration system that is described in this paper, is a generic system that can be trained to predictively transliterate between any two alphabet-based languages. Any new language can be supported within a matter of days. The addition of a new language involves defining the basic mappings between the input and the output language, which in theory can be as simple as specifying all the possible ways of representing an output character in input alphabet. However, context specific modelling is supported to have more control on incorporating language specific knowledge. The key idea here is to use thus defined simple language model to generate transliteration parallel corpus for training the prediction module. An AJAX based, rich text editor using the Quillpad technology is available for free use on www.quillpad.in .

Quillpad is the leading solution for Indian language input method online. However, there are a few open issues that are not handled at present by the Quillpad system. Though the current system is very good for most practical scenarios, it does not predict colloquial language well. Since Indian languages are generally phonetic, multiple different dialects and phonetically similar versions of a given word are accepted in written form. This, while rules out dictionary based approaches (See the blog entry for details: http://blog.tachyon.in/2012/03/02/does-quillpad-use-dictionary-for-prediction/), also poses another challenge. Most of the available corpus does not contain the colloquial word forms. Thus, for such words the transliteration would depend only on the generalisation performance of the system. Since a model is not learned on such patterns, the prediction quality is affected. However, it is possible to design a better learning algorithm that can learn to predict the colloquial forms effectively as colloquial forms themselves are some variants of the dictionary form of the word. And the transformation seems to depend mostly on the constraints introduced by our speech production or psycho-acoustic factors. It would be an interesting research topic, though we haven't felt the commercial need for it yet.

Another problem that is not effectively addressed in current Quillpad system is that of transliteration of English words inflected with Indian language suffixes. In South Indian languages, it is quite common to use English words attached with Indian language suffixes. However, the current way of handing Indian language prediction and English word prediction separately, is not the right approach for addressing this issue. A better approach is needed.

Finally, multiple words in Indian languages can be compounded into one word. This scenario is more common with South Indian languages. At the location where two words are joined, the consonants and vowels get replaced or skipped. These changes the local features used for predicting the class of certain characters. This might lead to error in predicted classes for those characters.