On Using Classical Poetry Structure for Indian Language Post-Processing

Anoop M. Namboodiri, P. J. Narayanan and C. V. Jawahar
International Institute of Information Technology, Hyderabad, INDIA
{anoop, pjn, jawahar}@iiit.ac.in

Abstract

Post-processors are critical to the performance of language recognizers like OCRs, speech recognizers, etc. Dictionary-based post-processing commonly employ either an algorithmic approach or a statistical approach. Other linguistic features are not exploited for this purpose. The language analysis is also largely limited to the prose form. This paper proposes a framework to use the rich metric and formal structure of classical poetic forms in Indian languages for post-processing a recognizer like an OCR engine. We show that the structure present in the form of the vrtta *and* prāsa *can be efficiently used to disambiguate some cases that may be difficult for an OCR. The approach is efficient, and complementary to other post-processing approaches and can be used in conjunction with them.*

1. Introduction

Post-processors are critically important to enhance the performance of language-recognizers such as OCRs, continuous speech recognizers, etc. They help to correct the errors in classification and provide more robust output from the language point of view. Post-processors are often language-specific and exploit the special features of the language to get high performance [5].

Post-processors rely on different aspects of a language. Dictionary-based post-processors use the vocabulary of a language to correct recognizer errors. Approaches for efficient dictionary lookup such as *Tries* [3, 7] has been explored. Other post-processors use deeper language rules to validate a given word in its context [2, 8], using morphological analysis. Script structure has also been modeled for recognition [11]. Statistical approaches use word-level and character-level joint probabilities (n-grams), for post-processing [12]. A combination of all such information needs to be used to correct for errors in individual methods.

Work on OCRs and post-processing have concentrated on *prose* due to its wider interest and greater prevalence. However, poetry or verse need to be handled for a complete treatment of the system. In languages like English, the word-level recognition may be largely the same for prose and poetry, except for the morphological analysis and the joint-probabilities. The problem is more serious in Indian languages, which have a much richer tradition in poetry or verse than European languages.

The *sandhi* rules of Indian languages allow words to be combined into longer strings of words. This makes recognition much difficult as word segmentation is not straightforward. This is further complicated in poetry as the word ordering is not as strict as in the prose form. For a recognizer of the written form like an OCR, further complications arise due to the closeness of the pictorial forms of different letters.

Post-processors using the state-of-the-art in other languages have been designed for Indian languages also [13]. Dictionary based post-processing using a vocabulary [14] and statistical models of words [4] has been applied to Indian languages. The complexity of the Indian language models have seriously limited the application of effective post-processors. Interestingly, the classical Indian literature – which stems from the Sanskrit literature – has strong computational bias [9] and have been subject to many language processing applications including language translation.

In this paper, we present the use of the metric information from classical Indian poetry structure as an effective clue for post-processing the output of an OCR. In particular, we show how the *laghu* and *guru* classification of aksharas and their conjoining into well-structured meters or vrttas can be used to overcome errors in the classification of the underlying components. Though we present it from the point of view of an OCR, these observations are relevant to other recognition tasks involving Indian languages, including continuous speech recognition and language parsing. To our knowledge, this is the first use of the rules specific to the verse form of any language for post-processing of recognition output. We show two approaches based on the vrtta structure: a string matching approach and a statistical approach. This paper presents the framework and points the way towards the use of the structure. The structural knowledge may be exploited in many other ways also.

कश्चित्कान्ताविरहगुरुणा स्वाधिकारात्प्रमत्तः
शापेनास्तंगमितमहिमा वर्षभोग्येणभर्तुः
यक्षश्चक्रे जनकतनयास्नानपुण्योदकेषु
स्निग्धच्छायातरुषु वसतिं रामगिर्याश्रमेषु ।।

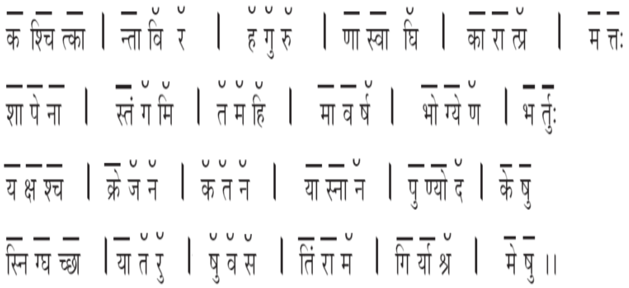


Figure 1. A shloka from *Mēghasandēsham* by *Kālidāsa*, and the same marked with the akshara types.

2. Structure of Ancient Indian Poetry

The *vēdās* (there are four, namely, *rk, yajus, sāma, atharva*) are poetic forms composed several thousand years ago; some from 4000 BC or earlier (Other opinions exist about their exact times, with some placing them one or two millennia later.) The compositions were transmitted from one generation to next orally as writing came much later in history. The poetic form had total dominance in the oral tradition due to the ease of committing verses to memory. Strict meter in the form of *chhandas* or *vrtta* and repetition of sounds (like alliteration) in the form of *prāsas* are prevalent in the Indian poetic form to aid oral transmission. The vrtta rules enforce a rhythmic pattern to the verses.

2.1. Vrtta: Strict Metric Structure

The chhandas or vrtta are rules of poetry defines a set of structures or frames for composing the lines of a poem, and is similar to metre in English poetry (closer to Greek and Latin poetry). Each vrtta defines a sequence of syllable types for each line of a poem so that the poem follows a rhythm when read out. The poet can choose to compose his work in any accepted vrtta.

However, in order to stick to the vrtta rules, poets make liberal use of the flexibility in word ordering in languages like Sanskrit. Moreover, the sandhi rules are extensively used, giving rise to long words formed by combining multiple root words. For example, in the verse in Figure1, The first word is a combination of four root words, while the second word of the third line is formed from five different root words. These characteristics make the use of both word n-grams and dictionaries extremely difficult in post-processing.

The basic linguistic unit of an *akshara* is a fusion (and not a succession) of 0 to 3 consonants and one vowel with an optional vowel-modifier. There are about 35 consonants, 15 vowels, and 2 or more vowel modifiers (*anuswāram, visargam,* and *chillu* in some languages). To define the vrttas, each akshara or syllable of a word is classified as either a *laghu* (∪) or a *guru* (–). Laghus are those aksharas that are short and soft, while long or hard sounding aksharas are classified as guru. Some of these concepts are observed in Greek and Latin poetry also. The grammar of poetry defines a set of rules that can classify each akshara in a word as a guru or laghu. Section 3 gives the outline of an algorithm to do this classification. Each vrtta is then defined as a sequence of laghus and gurus in groups of 1, 2 or 4 lines. The sequence repeats thereafter. For example, the vrtta named *Indravajra* has the following sequence for each line: – – ∪ – – ∪ ∪ – ∪ – –. Every shloka strictly follows one of the vrttas in classical Indian poetry. A rare exception is the use of a short akshara in place of a long one at the end of a line which is stretched while reading, to keep the rhythm. Figure 1 gives an example of a verse from *Mēghasandēsham* by *Kālidāsa*, along with the classification of each akshara. It is customary to form groups of three syllables for each line of the poetry, except at the end. Note that each line follows the same akshara pattern that is specified by the vrtta, *Mandākrānta*.

The rules governing metres in poetry have been the subject of much study in the Sanskrit tradition from more than three thousand years ago. A looser form of the chhandas is used as a rule in the vēdās that date back to 4000 BC or earlier. The theory of the stricter vrtta form appears in classical treatise on Sanskrit poetry structures like *Chhandas Shastra* and *Vrttaratnākaram* [10, 6], which date back several centuries before the start of the common era. Modern treatises on the topic are available in several Indian languages such as the *Vrttamanjari* by A. R. Rajaraja Varma [1] that was published as late as 1904.

2.2. Prāsas: The Optional Structure

In addition to vrttas, which are mandatory rules, there are many optional structures that are commonly used in traditional Indian poetry to enhance their beauty. *Prāsas* refer to repetitive sounds in a line or successive lines of a verse. This is similar to the concept of *Rhyme* in English poetry, where the last syllable is often repeated in consecutive or alternate lines to make the poem sound pleasing. In English, similar sounding syllables can have very different spellings (e.g., puff and enough), and hence cannot be used directly for post processing. However, for phonetic alphabets as used in Indian languages, similar sounds also means the same akshara, and hence can give a strong clue for recognition.

There are a variety of prāsas used in classical Indian poetry. For example, the *dwitīyākshara prāsa* is commonly used by poets, where the second akshara in each line in a verse is made the same. *Antyaprāsa* is the common form of rhyme used in English, where the last syllable of a line repeats in consecutive lines. *Ādyaprāsa* refer to the repetition of the first akshara of each line. The presence of such prāsas in a poem can give additional cues when the classifier output is ambiguous. However, as the presence of prāsas is optional, they need to be incorporated in a probabilistic framework for recognition. Another structures that is useful is that of *yati* (or pause), which results in a word division at specific points. However, poets often break it by introducing composite words, and hence should be treated as an optional structure.

3. Modeling and Processing of Poetry

Using the structural information involves two steps: the conversion of a given verse into a sequence of laghu and guru aksharas and matching with the candidate vrtta. Since each letter or akshara is either laghu or guru, we can encode each as a binary bit with 0 denoting laghu. Each line of the candidate vrtta can also be written as a string.

3.1. Encoding the Mātra String

As explained in Section 2, the assignment of a letter into laghu or guru mātra is based on a strict set of rules. A letter is guru if the vowel is elongated, or a vowel modifier follows the letter, or the following akshara is a composite-letter of the form *ccv* or *cccv*. The following steps can perform the encoding.

1. Segment the text into aksharas.

2. For each akshara, check if it has (a) the elongated vowel form, (b) a vowel-modifier like anuswaram, visargam, or chillu immediate after it, or (c) if the next akshara is a composite one.

3. If any of the conditions is satisfied, label the akshara as a guru, else a laghu.

The procedure is straightforward except for the segmentation of the aksharas. Segmentation is a critical step in all recognition and is being developed by several researchers currently. The given verse is converted to a binary mātra string by the above process and can be compared with the string for the candidate vrtta.

3.2. Algorithmic Post-Processing

If the given verse belongs to the candidate vrtta, the mātra strings for both will be identical. If the recognizer that produced the akshara labels for the given verse makes mistakes, the string will not match. Conventional string-matching techniques can be used to verify the match. The *edit distance* between the mātra strings can give an rough estimate of the errors in recognition. This can be followed by a scan through the strings to identify discrepancies.

Several algorithmic approaches are possible at this stage. We can assume that the verse belongs to the candidate vrtta if the edit distance is small. The aksharas for which the mātra labels do not match can then be analyzed closely. If a laghu is found where a guru was expected (or vice versa), the modified symbol for the associated vowel may be mismatched. The classifier output can be examined again and corrected if known confusion exists between the pictorial forms. This method can correct a significant number of confusion between close forms. Figure 2 shows the pairs *ki/kī*, and *ku/kū* in Malayalam and Telugu. (The upper bar on a vowel symbol indicates its elongated form.) It is clear that the pictorial forms could use additional clues for accurate recognition.

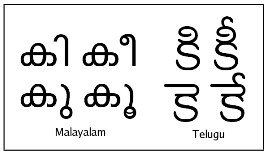


Figure 2. Pictorial forms of *ki/kī* (top), and *ku/kū* and *ke/kē*(bottom) in Malayalam and of Telugu. Note the similarity of glyphs.

3.3. Statistical Post-Processing

Post-processors are more effective when the uncertainty in classification is propagated up to the post-processor. This provides a statistical framework in which the classification can be studied. We present a presentation of the poetry structure in a statistical framework. Let the probability of a sequence of input glyphs g_i belonging to the akshara c_j be $p(c_j|g_i)$. This could be the posterior probability of the classifier.

Let $l(c_j) \in \{0, 1\}$ be the function that maps an akshara to laghu (value of 0) or guru (value of 1). This mapping is performed using the procedure described earlier. A verse consisting of aksharas $c_j, 1 \leq j \leq N_V$ belongs to a vrtta V if the string of $l(c_j), 1 \leq j \leq N_V$ matches withe mātra string of V, where N_V is the number of aksharas in vrtta V. For instance, the vrtta *Indravajra* would have a mātra string 11011001011.

Given a set $p(c_j|g_i), 1 \leq i \leq N$ of aksharas from a line of a shloka and their classifier outputs, the probability of a vrtta V is given by

$$P(V) = \prod_{i=1}^{N_V} f_V^i(c_j)\, p(c_j|g_i),$$

where $f_V^i(c)$ is an indicator function that matches the akshara c with the expected mātra from the vrtta. That is, $f_V^i(c) = l(c)$ if vrtta V has a guru at position i and $f_V^i(c) = (1 - l(c))$ if vrtta V has a laghu at position i. (That is, if $f_V(c_j)$ equals 1 if the corresponding akshara of V and c_j in their mātra and equals 0 otherwise.) If the vrtta is not available, we can estimate of the probability of each vrtta V, using dynamic time warping or by defining an HMM for each vrtta.

Once the vrtta is obtained, the problem is to find the best recognition alternative, that confirms with the mātra structure of the vrtta. We formulate this as a dynamic programming problem, on an $N_V \times K$ array, where K is the number of alternatives suggested for each akshara by the recognizer. Each cell in the array denotes the probability of the prefix of the akshara, matching the mātra structure. To compute this probability, we consider three factors: a) the probability of the akshara and its fitness to the mātra at the location (Pr_{ik}), b) the fitness of the changed class label of the previous akshara due to the current akshara ($C_{i-1,j,k} \in \{0, 1\}$), and c) the probability of the prefix string ($P_{i-1,j}$). Then the probability at a position is given by:

$$P_{ik} = \max_j \left(Pr_{ik} \times C_{i-1,j,k} \times P_{i-1,j}\right)$$

Finally, the value of $P(V)$ is obtained as the best alternative for the complete string from the last column of the array: $\max_k P_{N_V,k}$. The computation is illustrated in Figure 3, and is similar to Trellis decoding, employed in HMMs. A decoding step can recover the aksharas $c_i, 0 \leq i \leq N_V$ that is the corrected version of the verse by traversing the most likely path. The computations can be done efficiently.

Modeling Prāsa: In addition to the above cases, where the recognition result is clearly not allowed by the vrtta rules, our framework can also handle probabilistic rules as mentioned by Prasas. This is directly incorporated into the recognition output, where the probabilities of each akshara in the first, second, or last positions are modified in case of the presence of ādyākshara, dwithīyākshara or anthyākshara prasas.

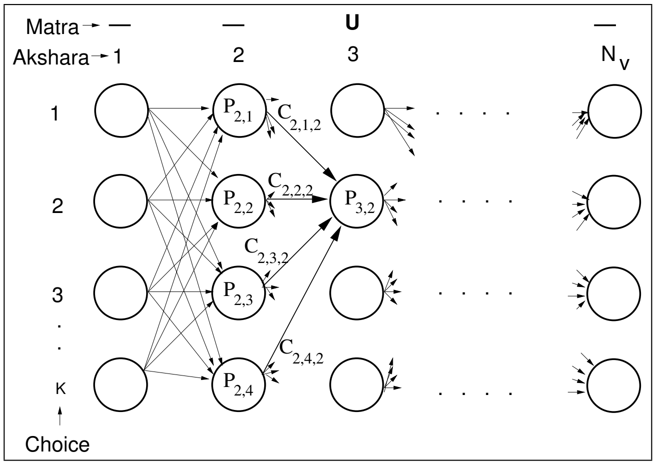


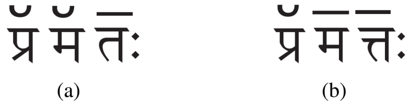
Figure 3. Decoding of the most probable akshara sequence.

3.4. Validation and Discussion

To ascertain the effectiveness of the above algorithm, we have tried it on a variety of vrttas and languages. We illustrate the working of the algorithm with an example from the verse in Figure 1. Consider the last word in the first line of the above verse, *Pramatta*.

Table 1. Recognition alternatives of the word in Fig. 4(a) along with their probabilities.

Input	Rank	Output	Prob.	Type	Class
प्र	1	pra	0.62	ccv	∪
	2	pa	0.21	cv	∪
	3	ma	0.17	cv	∪
म	1	ma	0.45	cv	∪
	2	pa	0.43	cv	∪
	3	ya	0.12	cv	∪
त्तः	1	ta:	0.46	cvm	–
	2	tta:	0.41	ccvm	–
	3	tra:	0.13	cvm	–



(a) (b)

Figure 4. (a) recognition result of the last word in line 1 of Fig.1, and (b) the corrected word.

The recognition result of the last word of line 1 (Fig.1) is shown in Figure 4(a). Note that the last letter, *tta*, was incorrectly recognized by the OCR as *ta* (the softer version) as the glyphs are very similar in shape. Consider the recognition results of the word given in Table 1. The table gives the top three recognition candidates, as per the akshara recognizer and the corresponding akshara types and classes (the *m* in the akshara type indicates a modifier). Computing the overall probability, the best choice is given by the result in Figure 4(a). The string pattern corresponding to the recognition result is $\cup \cup -$, while the one specified by the vrtta is $\cup - -$. Clearly, there is an error in the recognition result. The second choice would be the option $Ranks : 1, 2, 1$ for the three aksharas from Table 1. However, this also will not satisfy the vrtta. The third choice, $Ranks : 1, 1, 2$ would satisfy the requirement as per rule 2(c) in Section 3.1 (laghu becomes guru if followed by cc*). Hence our post-processor would select this choice and the result is shown in Figure 4(b).

Computational Complexity: In languages such as Sanskrit, where multiple words can be combined using sandhi rules, the vocabulary size is exponentially large in terms of the root words. The primary advantage of this post-processing method is that it is completely independent of the size of the vocabulary or a corpus. Let N be the number of aksharas in a line of the verse (typically, $N \in [8, 21]$), and let k be the number of choices output by the recognizer for each akshara (typically, $k \in [1, 5]$). To determine the best alternative for each akshara, we need to examine the alternatives for the previous, as well as the next akshara. Using the dynamic time warping formulation, we can compute the most likely sequence of aksharas that satisfy the given vrtta in $O(k^2.N)$ time. This includes the classification of the candidate aksharas into guru or laghu. In case the vrtta is not available, we need to estimate the most likely vrtta. This classification would take $O(v.N)$ time, where v is the set of possible vrttas. Although the set of available vrttas is around 320 [1], the popular ones are only around 25, and we can further restrict our search based on the number of aksharas in each line (typically, $v \in [3, 10]$). The overall time complexity of the algorithm is hence $O(N(k^2 + v))$.

Thus, post-processing is light, both computationally and in terms of the data requirements. In addition, the algorithm can work in conjunction with other post-processing steps such as dictionary lookup or character n-gram probabilities. It is also interesting to note that the approach is independent of the language also, as long as it follows the poetic structure as most Indian languages do. If the recognition output is encoded as either UNICODE or ISCII, the classification of aksharas into guru and laghu and the subsequent post processing is generic and applicable to any Indian language.

The pictorial forms of many of aksharas are hard to distinguish using standard pattern recognition techniques applied to the images. Similar symbols often result from using different vowel modifiers on the same consonant as in Figure 2. However, these are the easiest to handle for our post-processor, since a change in the modifier often changes the aksahra class, and hence the vrtta structure.

4. Conclusions and Comments

In this paper, we have presented a framework for incorporating the metric information present in classical Indian poetry to enhance the results of recognition. The information is relatively easy to encode, and can be used very efficiently to improve the recognition results. The approach is independent of the vocabulary size and even the language, provided the verses follow the metre structure. The algorithm can be used in conjunction with other post processing approaches and is effective in correcting modifier symbols, which are difficult to recognize for an OCR. The work can also be extended to the additional vrttas used in certain regional languages, that are not as rigid as the classical Sanskrit vrttas.

References

[1] A.R.RajarajaVarma. *VrttaManjari*. National Book Stall, 1904.

[2] Dan Klein and Christopher D. Manning. Natural Language Grammar Induction with a Generative Constituent-context Model. *Pattern Recognition*, 38(9):1407–1419, Sept. 2005.

[3] Edward Fredkin. Trie Memory. *Communications of the ACM*, 3(9):490–499, 1960.

[4] G.S. Lehal and Chandan Singh. A post-processor for Gurmukhi OCR. *Sādhana*, 27(1):99–111, Feb. 2002.

[5] Henry S.Baird. Anatomy of a Versatile Page Reader. *Proceedings of the IEEE, Special Issue on Optical Character Recognition(OCR)*, 80(7):1059–1065, 1992.

[6] Kedarabhatta. *Vṙttaratnākara*. c. 800 AD.

[7] D. Knuth. *The Art of Computer Programming, Volume 3: Sorting and Searching*. Addison-Wesley, 3rd edition, 1997.

[8] Li Zhuang and Xiaoyan Zhu. An OCR Post-processing Approach Based on Multi-knowledge. In *Proc. KES*, pages 346–352, 2005.

[9] Pānini. *Ashtādhyāyi*. c. 500 BC.

[10] Pingala. *Chhandas Sastra*. c. 200 BC.

[11] R.M.K. Sinha. PLANG-A picture language schema for a class of pictures. *Pattern Recognition*, 16(4):373–383, 1983.

[12] Tin Kam Ho and George Nagy. Exploration of Contextual Constraints for Character Pre-Classification. In *Proc. ICDAR*, pages 450–454, Sept. 2001.

[13] U Pal and B B Chaudhuri. Indian Script Character Recognition: A Survey. *Pattern Recognition*, 37:1887–1899, 2004.

[14] V. Bansal and R.M.K. Sinha. Partitioning and Searching Dictionary for Correction of Optically-Read Devanagari Character Strings. In *Proc. ICDAR*, pages 653–656, 1999.