2009 10th International Conference on Document Analysis and Recognition

Shape Encoded Post Processing of Gurmukhi OCR

Dharam Veer Sharma[1] Gurpreet Singh Lehal[2] Sarita Mehta[3]
Department of Computer Science, Punjabi University, Patiala, INDIA
dveer72@hotmail.com[1], gslehal@gmail.com[2], mehta.sarita@rediffmail.com[3]

Abstract

A post-processor is an integral part of any OCR system. This paper proposes a method for detection and correction of errors in recognition results of handwritten and machine printed Gurmukhi OCR. Based on the shape similarity of characters, the consonants of Gurmukhi Script are divided into different sets. Each set is given a unique number. In case of a recognition error, based on the shape of the consonants, corrections are made by taking each consonant of the subset into consideration. According to proposed algorithm, each recognized word is first encoded based on its consonants. The corresponding code is then searched in the dictionary. If it exits then words from the list of the code are match with the source word. In case of match the word is treated as correct else suggestions are made based on the similarity of the source word with the words of the same code present in dictionary. The method has been tested on the output of OCR of variety of machine printed and handwritten documents.

1. Introduction

In case of handwritten text, OCR systems are often not able to return correct data because writing style of the writers or limited capabilities of the system itself. The recognition process returns a sequence of characters to be verified and replaced, if needed, by a post processor, in order to ensure the correctness of the sequence and extract meaningful words. Deviation in recognition results may be because of structural similarities of some characters, noise or degradation of source text, machine printed or handwritten.

The meanings of the words shown in figure 1 are quite different but structurally the words look similar. Due to large variation in handwriting styles words given in figure 1 can confuse the occurrence of one with the other. The recognition process may have identified the word differently, whether given word is ਚੋਰ (CHOR) or ਚੌਰ (CHAUR) in figure 1(a) and similarly whether ਸੇਵਾ (SEVA) or ਮੇਵਾ (MEVA) in figure 1(b), both words in both cases have very similar structure but the meaning of the words are totally different.

ਚੋਰ ਚੌਰ	ਸੇਵਾ ਮੇਵਾ
a	**b**

Figure 1. Two sets of similar looking words

Correct recognition of structurally similar characters is a difficult task. OCR task is not over at recognition itself. There has to be some system for validation of the recognition results, which can check the correctness of the results of recognition process. And in case of wrong recognition, it should be able to correct or atleast suggest some correction. Use of such a system further improves the overall recognition rate.

The objective of this research work is to develop a shape similarity based post processing system for machine printed and handwritten Gurmukhi Script. The system resolves the ambiguity problem and decreases the recognition error rate. The system can also be used for post processing of form recognition system.

The rest of the paper has been organized as follows: section 2 discusses some existing post processing techniques; section 3 covers the proposed research work, experimental results are given in section 4, the research work has been concluded in section 5 and references are given in section 6.

2. Previous work

Optical Character Recognition is useful and important for its application potential in office automation, automatic data entry in banks, libraries and post offices, reading aid for blind and multimedia design etc. There exists extensive literature on Roman and Oriental OCR systems, but research on Indic scripts is still nascent. These are some OCR systems for Indian scripts like Gurmukhi[1], Devnagari[3] and Bangla[7].

Different post processing approaches have been used by researcher. Chaudhuri and Pal [4] have developed an OCR error detection and correction technique for highly inflectional language script like Bangla. The authors have used two separate lexicons of root words and suffixes, candidate root-suffix pairs of each input word are detected, their grammatical agreements are tested and the root/suffix part in which the error has occurred is noted. The correction is made on the corresponding error part of the input string by a fast dictionary access technique. To do so some alternative strings are generated for an erroneous word. Among the alternative strings, those satisfying grammatical agreement in root-suffix and also having smallest Levenstein distance are finally chosen as correct ones. The system has an accuracy of 75.61%.

Lehal et. al[2] developed a shape based post processor for Gurmukhi OCR. Based on the size and shape of a word, the Punjabi corpora were split into different partitions. The statistical information of Punjabi language syllables combination, corpora look up and holistic recognition of most commonly occurring words was combined to design the post processor. The corpus was partitioned into two levels: for first level the corpus was split into seven disjoint subsets based on the word length and at second level, shape of word was used to further segment the subsets into a list of visually similar words. For identification of visually similar words a set of robust, font- and character-size independent structural features were used. The recognition accuracy of the OCR without post processing was 94.35%, which was increased to 97.34% on applying the post-processor to the recognized text.

Bansal and Sinha [3] described a correction method for optically read Devanagari character strings which used a partitioned word dictionary. The partitioning of dictionary was done in order to reduce the search space besides preventing forced match to incorrect word. The envelope information of words consisting of number of top, lower, core modifiers along with number of core characters form the second level partitioning feature for short words partition and the remaining words were further partitioned using tags (a string of fixed length associated with each partition). The search process involved distance matrix for assigning penalty for a mismatch. An improvement of approximately 20% in the recognition performance was obtained.

There are two approaches for judging the correctness of the spelling of a word. The first approach estimates the likelihood of a spelling by its frequency of occurrence which is derived from the transition probabilities between characters. This requires a priori statistical knowledge of the language. In the other approach, the correctness is judged by consulting the dictionary. A hybrid approach attempts to combine the best of both the approaches by amalgamating them at an appropriate stage of processing.

The method proposed by Wing Seong wong et. al[8] is a novel approach for incorporating OCR technology within a form-processing environment in order to extract contextual cue words. The information is then used to provide contextual information to the Cursive Script Recognition (CSR) system so that different sets of lexicons can be chosen for different fields in a form in order to produce an improved recognition rate. Contextual information is used because a form consist of form frames, pre-printed data and user filled-in data and these 3 elements are closely related to each other in that the lines and boxes usually signify the filled-in data areas, whilst the machine printed text that is adjacent to lines and box areas are normally the guiding text that specifies what the filled data should be. The recognition rate for the proposed method has an overall improvement of 12% (from 43% to 55%) as compared to a form processing system that has not utilized the contextual information.

Hyuk-Chul Kwon[9] has developed a post processor for contextual post processing of a Korean OCR system using linguistic constraints. Authors focused on the contextual post processing by selecting the most feasible word from multiple output strings of an OCR system. The correction is applied only when the selection fails. After selecting a feasible word, it confirms the word by the linguistic constraints as collocations of words. The collocation between words is applied for short words especially one, two and three syllable words because such words are frequently mis-selected. The system improved the word recognition rate of OCR system from 90.50% to 94.72%.

The scheme developed by Shang-Lin Hseih[10] for rectifying recognition results of printed Chinese characters, aimed at rectifying the error, that occurred when the correct character was not the highest ranked character in a candidate set or was excluded from the set, in printed Chinese character recognition. The scheme applies word information and line-segment cancellation to rectify recognition errors of a classifier. The scheme is not only able to pick out the correct character not in the highest rank position but can also recover the correct character originally excluded from a candidate set. Authors have claimed that 75% of the recognition errors for normal quality documents and 48% for low quality ones were rectified by their rectification scheme.

3. Proposed solution

The objective of the post processing is to correct errors or resolve ambiguities in the OCR results by using various methods. As shown by Lehal and Singh

[2] an improvement of 3% in the recognition rate from 94.35% to 97.34% has been reported on machine printed text using the post processing techniques developed by them. The technique presented is not limited to machine printed text of Gurmukhi script; it has been effectively applied on the handwritten text of Gurmukhi script as well.

As shown in Figure 1, due to varying handwriting styles, words which are semantically very different appear to be the same structurally. In these types of cases it is very difficult for the OCR to correctly recognize the relevant word. The proposed method solves the ambiguity problem up to a certain level by using shape similarity of the Gurmukhi characters and by using dictionary of correct words.

The basic idea of the proposed scheme lies in dividing the alphabets of Gurmukhi script into 9 subsets based on their shape similarity. The characters are placed in same subset based on the shape similarity of the characters as the OCR may recognize one character in different ways e.g. ਪ and ਮ, ਖ and ਥ, ਮ and ਸ are very little different from each other. The problem becomes more complex in case of handwritten text. Following are the steps to perform post processing on the recognition results:

Step 1 Creation of subsets

Before applying the process on the recognition result, we need to define subsets of consonants based on their shape similarity and number each of the subsets. For Gurmukhi script 9 subsets have been identified as given in table 1. The consonants of Gurmukhi script are divided into 9 subsets based on their shape similarity.

Table 1: Subsets with corresponding codes

Subset Code	Subsets of Consonants
1	ੳ ੲ
2	ਚ ਟ ਰ ਹ ਢ ਦ ਫ ਫ਼ ਣ ਝ ਵ ਛ
3	ਜ ਜ਼
4	ਖ ਖ਼ ਬ ਧ ਥ ਪ ਯ ਮ ਸ ਸ਼
5	ਕ ਙ
6	ਗ ਗ਼
7	ਅ ਘ
8	ਤ ਡ ਭ ੜ
9	ਠ ਨ ਲ ਲ਼

The vowels have been ignored while creating the codes, but are taken into account when the word matching process is carried out (steps 3 and 4).

Each word read from OCR output is given a code depending upon subset number from the above table. The maximum size of the code has been restricted to 8 digits. Experimentally it has been observed that words with 8 or more consonants seldom cause ambiguity as they are significantly different from each other. For example word ਸਰਲ produces code: 429 which is padded with zeros to make it an 8 digit code thus resulting in 42900000.

Step 2 Creating initial dictionaries

The next step is to create dictionaries. Separate dictionary have been create for 73,808 words and 24,548 Indian names (including first, last and middle names) in Gurmukhi script. Unique words or names are first stored in an AVL tree from a file. Where each node of the AVL tree represents a unique code and is of the following form:

```
STRUCTURE NODE
    Code: Code generated for a word
    WordsList: List of words with same code
    LEFT*: left child pointer
    RIGTH*: Right child pointer
    BF: Balancing factor
End NODE
```

The AVL tree creation algorithm works via the following steps:

1. Generate code Wc of the word using the coding scheme given in table 1.
2. Search for the code Wc in the AVL tree.
3. If code is found in the AVL tree then add the corresponding word of the Wc to the word list in AVL tree.
4. If code is not found add a new node with Wc as code (perform rotations if necessary).

Finally, breadth first search is applied on the AVL tree and traversed nodes are stored in a dictionary file. This is done, so as to re-create the AVL tree without performing any rotation.

The AVL tree has two purposes, first to utilize its search efficiency, as its complexity is O($\log_2$N), where N is number of nodes in the tree, second for inserting a new word or name, which is not present in dictionary.

In the tree nodes are be created on the basis of codes. Thus the number of nodes is equal to the number of distinct codes generated. Corresponding to each node, there is a list of words that holds the words having same code. For example, corresponding to code

75900000, the list of words is: ਅਕਾਲਾ, ਅਕਾਲ, ਅਕੇਲਾ, ਅਕਿਲਾ, ਅਕੁਲ, ਅਕੰਨ, ਅਕੀਲ, ਅਕਿਲ.

Step 3 Validation

When post processing is to be applied, first the dictionary is loaded into an AVL tree. Then each word of the OCR output is matched with the contents of the dictionary stored in memory as an AVL tree. The following algorithm is applied for validation:

1. Let S be the source word which is to be validated
2. First generate code Wc of the word S using the shape based coding given in table 1
3. Based on code Wc of word S apply search in AVL tree. If a matching code is found in the AVL tree then the word S is compared with each word of the words arrays of the matched code, using linear search, and
   a. If a corresponding word is found in the words list of code Wc then the source word S could be a correctly recognized word.
   b. If, S is not found in the words list and is an incorrect word then suggestions are made (step 4) from the words in the list stored in AVL tree with the code C.
   c. If, S is not found in words list but word S is correct as per grammar rules of Gurmukhi script then the word is added to the words list of the code found in AVL tree (step 5).
4. If code C is not found in the tree but the word is correctly recognized then a new node is created in the AVL tree with the new code and word S is added to its words list (step 5).

Step 4 Proposing Suggestions

If the word to be searched is not found in the dictionary and is not correct according to the grammar rules of Gurmukhi script, then nearest possible suggestions are offered to correct the word on the basis of code of the word from the dictionary. For example, if searching for a word ਕਵਿਡਾ with code 52800000, the post processor searches for the node having same code as the code of the word to be searched. If the code is found, it checks the words array list corresponding to that node for the desired word. If the word is not there, then closest word in form of shape or structure is offered as suggestion. For example in case of ਕਵਿਡਾ there is a node in our dictionary having same code i.e. 52800000, with list of words: ਕਾਰਤੀ, ਕਾਰਤਾ, ਕੱਟੜ, ਕੱਟੜਾ, ਕਵਿਤਾ, ਕਏਤ, ਕੀਰਤ, ਕੀਰਤੀ, ਕਿਰਤ, ਕਿਰਿਤ, ਕਿਰਿਤਾ, ਕਿਰਿਤਾਂ, ਕੀਰਤੀ, ਕੋਵਿਤਾ, ਕਰਾਂਤੀ. The nearest matching words are ਕਵਿਤਾ, ਕੋਵਿਤਾ so these words are offered as suggestion to correct the word, because they have same code and almost similar structure.

Step 5 Adding Word to Dictionary

If the word is not found in the dictionary but is correct according to the grammar rules, then it checks for the code. If the code exists in the tree but the word is not there, then the word is added to the list of words with same code. But, if the code is not found in the tree, then a new node with the new code is added and then for the new node the corresponding word is added to the words list. This may require rotations for balancing the AVL tree.

At the end of post processing, if new words have been added to the AVL tree then the tree is traversed breadth wise and stored in the dictionary file otherwise it is discarded and memory is freed.

4. Experimental results

In absence of any bench mark database, Gurmukhi OCR[1] recognition results were used for testing the algorithm. The initial statistics for words and names are given in table 2.

Table 2: Initial statistics for words and names

	Words	Names
Number of words in dictionary	73808	24548
Number of nodes formed in AVL tree	8453 (N)	2189 (N)
Minimum number of words corresponding to one code	1	1
Maximum number of words corresponding to a code	278	48
Average words per code	8.73	11.21
Search complexity for searching a code in worst case	13 O($\log_2$ N)	11 O($\log_2$ N)
Search Complexity for searching a word/name in words/names list of a code consisting W words/names	W O(W)	

In all, the algorithm was applied on OCR output of 48 documents with average 431 words per document, before applying post processing proposed by G. S. Lehal et. al[2]. The average recognition accuracy improved by 4.65 per cent.

The words and names dictionaries comprised of 73,808 words and 24548 names arranged in 8453 and 2189 nodes respectively of the AVL trees. Average number of words and name per node was 8.73 and 11.21 respectively. The frequency distribution of number of words/names is given in table 3.

Table 3: Frequency distribution of number of words/names per node

No. of Words/Names Per Node	No. of Nodes for Words	No. of Nodes for Names
0-50	8039	2089
51-100	377	92
101-150	14	6
151-200	13	2
200-250	8	0
250-300	2	0
TOTAL	8453	2189

The algorithm has also been tested on output of form processing system for Gurmukhi script. A total of 500 forms were processing and recognition output subjected to the post processor. Primarily names were validated, which resulted in 7.45 per cent improvement in recognition accuracy.

5. Conclusion

The algorithm is simple to implement with high speed of post processing. It does not require any prior statistical knowledge of the language. A basic list of word is required to create dictionary. The accuracy obtained for OCR result improvement was between 4.65% for machine printed words and names and 7.45% for names obtained from form processing.

Just by modifying the coding scheme, this algorithm can be applied to other scripts as well.

6. References

[1] G. S. Lehal and Chandan Singh, "A Gurmukhi script recognition system", in Proceedings of 15th International conference on Pattern Recognition, vol 2 pp 557-560, 2000.

[2] G. S. Lehal, Chandan Singh and Ritu Lehal, "A Shape based post processor for Gurmukhi OCR", in Proceedings of 6th International Conference on Document Analysis and Recognition, Seattle, USA, IEEE Computer Society Press, USA, pp. 1105-1109, 2001.

[3] V. Bansal and R. M. K. Sinha, "Partitioning and searching dictionary for correction of optically read Devnagri character strings", in Proceedings of 5th International Conference on Document Analysis and Recognition, pp 653-656, 1999.

[4] B. B. Chaudhary and U. Pal, "OCR error detection and correction of Inflectional Indian Language Script", in Proceedings of 13th ICPR, pp 245-249, 1996.

[5] Bidyut Baran Chaudhary, "Towards Indian Language Spell-Checker design", in Proceedings of Language Engineering Conference, 2002.

[6] R. M. K. Sinha, "Rule based contextual post processing system Devnagri text Recognition System", Pattern Recognition, vol 20 pp, 475-485, 1985.

[7] B. B. Chaudhary and U. Pal, "A complete printed Bangla OCR system", Pattern Recognition, vol 31, pp 531-549, 1998.

[8] Wing Seong Wong, "Contextual Focus for Improved Recognition of Hand-Filled Forms", in Proceedings of 6th International Conference on Document Analysis and Recognition, 2001

[9] Hyuk-Chul Kwon, "Contextual Post processing of a Korean OCR System by Linguistic Constraints", in Proceedings of 3rd International Conference on Document Analysis and Recognition, pp 557-562, 1995.

[10] Shang-Lin Hseieh, "A new scheme for rectifying recognition results of printed Chinese characters", Pattern Recognition Vol 34, pp 2121-2132, 2001.