A Bilingual OCR for Hindi-Telugu Documents and its Applications

C. V. Jawahar, M. N. S. S. K. Pavan Kumar, S. S. Ravi Kiran
Centre for Visual Information Technology
International Institute of Information Technology
Gachibowli, Hyderabad, India - 500 019
jawahar@iiit.net

Abstract

This paper describes the character recognition process from printed documents containing Hindi and Telugu text. Hindi and Telugu are among the most popular languages in India. The bilingual recognizer is based on Principal Component Analysis followed by support vector classification. This attains an overall accuracy of approximately 96.7%. Extensive experimentation is carried out on an independent test set of approximately 200000 characters. Applications based on this OCR are sketched.

1. Introduction

Recognition of characters from document images is at the heart of any document image understanding system [11]. The activities and results on Indian language OCR are grossly insufficient even today. Recent spur in Indian Language OCR research is reflected in special sessions and increased number of contributions in ICDAR2001, ICDAR1999, ICVGIP2002, ICVGIP2000 and NCDAR. Focused documentation of the activities in this area may also be seen at [1, 3, 5]. These technical articles reveal the academic contributions to the development of OCR systems for Indian scripts and discuss specific issues to be addressed in Indian scripts. A major contribution in this area is the Devanagari OCR system developed by B. B. Chaudhuri of Indian Statistical Institute [6], which is commercially available. It is our view that an OCR engine has a much better impact as part of real-life applications than along with word processors. With the grass-root spread of computers, time is ripe in India for widespread applications based on character recognition in both rural and urban societies. The paper reports some of our efforts to achieve this.

In this paper, we describe a bilingual OCR for recognizing printed Hindi and Telugu texts. Hindi uses the Devanagari script [1]. In addition to Hindi, languages like Sanskrit, Marathi and Konkani use the Devanagari script, making it the most widely used script in India. However, real-life applications specific to a language need language specific tools. From the recognition point of view, Hindi and Devanagari can be considered synonymous. Hindi is the most popular language in India. Telugu is the second most popular language in India. These two languages are selected based on many other considerations. They represent two major classes of scripts in India – scripts with and without *sirorekha*(a head bar). Languages like Bangla, Oriya, Gurumukhi etc. are in the first class while Malayalam, Kannada, Tamil etc. are in the second. As pointed out in [12], OCR activities in Telugu are practically non-existent. Refer to their article [12] to find a discussion on Telugu alphabets and character composition rules.

Printed Devanagari character recognition has been attempted based on KNN and Neural Networks [5, 1, 6]. These results were extended to Bangla [6], which also has the *sirorekha*. Structural features like concavities and intersections were used as features. A similar approach was tried for Gujarati [4] with limited success. Reasonable results are reported for Gurumukhi script [5]. Preliminary results were also reported on recognition of two popular scripts in south India – Tamil and Kannada [3]. An attempt to recognize Telugu script using KNN and Fringe distance is reported in [12]

In this paper, we describe a bilingual OCR system and sketch some real-life applications that the system has been used for. Section 2 gives the component extraction procedure for the scripts under consideration. This is followed by the technical details of the features and classifiers employed in the proposed system in section 3. Section 4 describes the details and results of the experiments we carried out to evaluate the performance. Finally, applications and extensions of the proposed OCR is given in section 5.

2. Bilingual OCR

Indian scripts are phonetic in nature and syllables form their basic units. A syllable may be formed by composition of a few consonant and vowel sounds. Most of the time, vowels appear as vowel modifiers, i.e., they deform the consonant shape to articulate the composite sound. This creates the need for splitting the syllables into their constituent components which is a nontrivial task. Documents in India commonly contain multiple scripts. English, Hindi (the national language) and a regional language are used in most regions of the nation, which has more than 18 recognized languages and 15 scripts. Though there is a representation standard for Indian scripts (ISCII), none of the popular fonts are ISCII compliant. The glyphs and the combination rules differ from one font to another. This further complicates the recognition systems for printed text.

If one considers all the possible characters which can be generated by composition of consonants and vowels, a prohibitively large number of classes will result. For both Hindi and Telugu, the number of classes then become more than 10000. Interestingly, for Hindi, most of the modifiers (*matras*) appear either above the *sirorekha* or below the character. We can segment out such modifiers to reduce the number of classes to less than 100. However, composite characters (*samyuktakshars*) created by combination of up to three consonants and two vowels will have to be handled in a different manner. They are usually formed by placing a 'half character' next to a full character. We employ a separate classifier to recognize the half characters and combine the results from both the classifiers to recognize composite characters. For Telugu, we split the characters into constituent components such that the number of classes become manageable. This segmentation and the recognition of the resulting components is more complex for Telugu compared to Hindi. Telugu has about 330 distinct components while the number of full components are about 100 for Hindi.

2.1. Component Extraction for Recognition

Our OCR system scans document pages from a flat-bed scanner and does preprocessing to improve the performance of the recognition module. The scanned document is filtered, binarized and skew corrected before individual components are extracted. Projection profiles have been used for the skew correction in the range of $\pm 20°$. Details of the individual steps may be seen at [7]

For the languages under consideration, composite characters are minor modifications of the basic characters. We can decompose the character shapes into appropriate components for recognition and recompose the recognized components to determine the characters. Text blocks in the document pages are extracted first and line and word segmentation is carried out. Projection profiles, space between words and lines are used to achieve this [7].

Next, the script of each word is identified. To distinguish between Hindi and Telugu, *sirorekha* has been primarily employed. This is also supported with the context information. If the previous word is Hindi, next word is also in Hindi unless a strong clue is achieved to believe the other way. Next, individual words are processed to obtain the components for the scripts. For Hindi, segmentation involves the removal of sirorekha. Figure 1 shows the constituent components getting separated after the removal of the sirorekha (the horizontal bar) from a word. For Telugu, component extraction implies the separation of connected components. The individual connected components in Telugu are not distinct letters. They can also be modifiers. This results in splitting of the single word into many separated components.

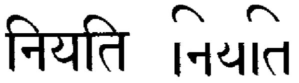
Figure 1. Sirorekha removal for a Hindi word

Once the *sirorekha* is removed, for Hindi, the top, middle and bottom zones are identified easily. Top zone gets automatically separated with the removal of sirorekha. Bottom zone is identified from the projections. Components in top and bottom zones for Hindi are part of vowel modifiers. Each of these components are then scaled to a standard size before feature extraction and classification.

3. Feature Extraction and Classification

Feature extraction is the identification of appropriate measures to characterize the component images distinctly. There are many popular methods to extract features. Considering the entire image as the feature is at one end of the spectrum. This representation is bulky and contains redundant information. At the other extreme, there are feature extraction schemes which consider some selected moments or other shape measurements as the features. Area, projection parameters, moments, fringe measures, number of zero crossings etc. are popular for recognition of Indian language scripts [3]. Such subjective geometric features provide reasonable performance for specific font and script. Our preliminary evaluation suggested that using the entire image as a feature vector provides excellent results for printed character recognition. However, the dimensionality is very high and the computational and memory requirements become serious bottleneck for applications. We use principal components for dimensionality reduction. This well known transformation is very popular in applications such as face recognition. Principal components can give superior performance for (a) font-independent OCRs, (b) easy adaptation across languages, and (c) scope for extension to handwritten documents.

Consider the ith image sample represented as a M-

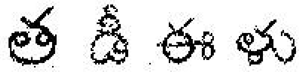
Figure 2. Some of the degraded Telugu characters

dimensional (column) vector $\mathbf{x}_i$, where M depends on the image size. From the large sets of training components, $\mathbf{x_1}, \ldots, \mathbf{x_N}$, we compute the covariance matrix as

$$\Sigma = \frac{1}{N}\sum_{i=1}^{N}[\mathbf{x_i} - \mu][\mathbf{x_i} - \mu]^T.$$

The data set practically lies in a lower dimensional space $K(<< M)$. We identify the required minimal dimension K such that $\sum_{i=1}^{K} \lambda_i / Trace(\Sigma)$ is 95% where λ_i is the ith largest eigen value of the covariance matrix Σ. Eigen vectors corresponding to the largest K eigen values are arranged as rows in $\mathbf{A}$ and a new feature vector is computed by projecting as

$$\mathbf{y}_i = \mathbf{A}\mathbf{x}_i$$

This lower dimensional ($K \times 1$) representation of the component images are used as the features for recognition.

SVMs for Classification K-nearest neighbor and Neural Network classifiers are popular for character recognition applications, especially for Indian scripts. Superior classification is expected to warrant large computational power. Storage capacity and computational power of personal computers have exploded in the last five years and it is now possible to employ better classifiers to achieve improved performance.

Support Vector Machines [8, 10] are pair-wise discriminating classifiers with the ability to identify the decision boundary with maximal margin. Maximal margin results in better generalization – a highly desirable property for a classifier to perform well on a novel data set. From the statistical learning theory point of view, they are less complex (smaller VC dimension) and perform better (lower actual error) with limited training data. Classifiers like KNN provide excellent results (very low empirical error) on a data set on which they are trained, while the capability to generalize on a new data set depends on the labeled samples used. Computational complexity of KNN increases with more and more labeled samples.

Identification of the optimal hyperplane for separation involves maximization of an appropriate objective function. The training process results in the identification of a set of important labeled support vectors x_i and a set of coefficients α_i. Support vectors are the samples near the decision boundary and most difficult to classify. They have class labels $y_i(i.e., \pm 1)$. The decision is made from $f(x) = sgn(\sum_{i=1}^{l} \alpha_i y_i K(x_i, x))$. The function K in the previous equation is called the kernel function. It is defined as $K(x,y) = \phi(x).\phi(y)$, where $\phi : R^d \longrightarrow H$ maps the data points in d dimensions to a higher dimensional (possibly infinite dimensional) space H. For a linear SVM, $K(x,y) = x.y$. We do not need to know the values of the images of the data points in H to solve the problem in H. By finding specific cases of Kernel functions, we can arrive at Neural networks or Radial Basis Functions. From the pairwise SVM classifiers, a Directed Acyclic Graph based multi-class classifier is formed [8].

The advantage of SVM classifier over other classifiers can be explained as follows. An Indian language OCR system generally has large number of classes and high dimensional feature vectors. Variability of characters is also very high at each occurrence. SVMs are well suited for such problems since they have excellent generalization capability. They always result in a global solution of the problem [10].

4. Experimental Results

We evaluated the performance of the recognition process on approximately 200000 sample characters for Telugu and Hindi. Most of these characters are collected in a systematic manner from printed pages scanned on a HP7670 Scanjet scanner. Around 25% of them were generated using a degradation model [9]. Documents were skew corrected and components were extracted. Subsets from this data set were used for training and testing in the experiments described in this section. We have observed that Hindi characters provide good recognition even at a resolution of 15×15 while Telugu characters need more details and work well with 40×40. After dimensionality reduction(PCA), Telugu needs around 150 features for representation, while Hindi needs only 50. We are presently working on alternate dimensionality reduction techniques.

In the first experiment, we consider the font-specific performance of Hindi and Telugu. We consider two fonts each for these languages – *Shusha* and *Nai Dunia* for Hindi, and *TL-TT Hemalatha* and *TL-TT-Harshapriya* for Telugu. These are among the popular fonts, even used by news papers. We consider an average of 20000 samples for the experiment. Results can be seen in Table 1. As can be seen from the top section of the Table 1, for the homogeneous data, recognition is very good for all combinations. Rare misclassifications were present due to the distortions created by the skew correction. KNN based classification results were computed for many values of K; $K = 5$ is shown in Table 1. SVM-based experiment was carried out for linear(SVML) and quadratic(SVMQ) kernels. For the entire image as feature vector, only linear kernel is reported.

Scale and resolution of characters usually influence the

Expt Details	Language	Font	KNN-Image	KNN-PCA	SVML-Image	SVML-PCA	SVMQ-PCA
Homogeneous Data	Hindi	Shusha	99.60	99.16	99.64	99.25	99.36
	Hindi	NaiDunia	99.51	99.20	99.86	99.13	98.96
	Telugu	Hemalatha	98.67	95.24	98.80	98.66	98.85
	Telugu	Harshapriya	99.30	99.03	99.25	99.67	99.02
Scale Independence	Hindi	Shusha	96.61	95.78	98.12	97.67	97.69
	Hindi	NaiDunia	96.11	94.30	97.00	97.12	97.17
	Telugu	Hemalatha	97.55	94.97	97.31	97.32	97.10
	Telugu	Harshapriya	96.22	92.73	97.12	97.22	97.13
Degraded Data	Hindi	Shusha	93.03	92.51	96.20	96.92	96.95
	Hindi	NaiDunia	94.60	93.40	96.50	96.98	97.06
	Telugu	Hemalatha	93.03	93.31	97.23	97.92	97.95
	Telugu	Harshapriya	93.10	93.93	97.14	97.83	97.72
Font Independence	Hindi	N. A	92.64	92.97	97.10	96.68	96.72
	Telugu	N. A	93.03	92.30	97.87	97.62	97.58

Table 1. Recognition accuracies of the system on synthetic and scanned data. The SVM results are shown for linear and quadratic kernels.

performance of the system. We considered samples printed at various sizes and scanned at various scales for the next experiment. This resulted in an average of 30000 labelled samples. Results are shown in Table 1. SVM-based classifiers are found to perform better than KNN. This result is better than the accuracies reported in [5, 12] on smaller or unknown data set.

Characters in a document image are usually degraded in quality. We use the degradation model given in [9] to simulate the distortions due to the imperfections in scanning. Black and white pixels are swapped according to some probabilities directly related to the distance from the boundary. Some of the degraded Telugu characters are shown in Figure 2. The results of degradation are given in Table 1. It can be observed that the degradation is more serious for Telugu than Hindi, for a fixed dimensional representation.

We considered around 90000 samples for the experiment on font independence. Hindi and Telugu are tested separately. It is found that the SVM classifiers perform well for the data set. For Telugu, SVM based results are more than 3% superior to that of KNN.

The results reported till now are component level accuracies. In real-life applications, one is interested in character level accuracies than the component level accuracies. Since characters and components are very similar in our formulation, character level results are comparable to the component level. We achieve 97.5% and 96.8% accuracies for Telugu and Hindi respectively. These results were computed from scanned document images. Telugu is found to give better result because of the larger image size and the number of principal components considered.

This system was tested on many printed documents. They include *suktam*s from the *Brahmasutra* – an authoritative text on Vedanta philosophy, pages of texts from an Indian language corpus [2] and many other sample test pages. Some of the pages were also bilingual. Based on the performance of the system on these test documents, results of font, scale independence and performance in the presence of degradation, we claim an overall accuracy of 96.7%. The paper and printing qualities were good for all these experiments. These results, as such, cannot be expected for newspaper and other real-life documents. Presently, we are working on algorithms to achieve good performance on real-life poor quality documents.

Support Vector Machines provide better results than KNN classifiers. Though we tried many kernels for the SVMs, we could not find any considerable difference between their performances. We use linear kernels for our applications. From the previous experiment, we find that the result of the SVM-PCA combination is computationally affordable and performance-wise superior. The proposed OCR has tremendous scope for further improvement. We are working presently in this direction. Without waiting to perfect the OCR accuracies, we want to take the OCR engine to applications where this accuracy is acceptable. These include a reading system for the visually impaired and a document database for indexing and retrieving. Very brief description of these extensions are provided in the next section.

There are also applications where a smaller number of features will be preferable. We are presently working on finding a set of optimal discriminant vectors for the pairwise classification using SVMs. Our preliminary results suggest Our preliminary results using optimal discriminant vectors suggest that a reduction in number of features to an order of 10 could retain comparable or better performance.

5. Applications and Extensions

Research and development in the area of printed Indian script recognition was taken up by our group to address many specific and generic applications which need an OCR engine. The system is capable of accepting document images from a file or directly from a scanner. Recognized text is displayed and could be edited. These add-on features are well separated from the core OCR engine based on Support Vector Machines. Our strength is in extending the recognizer to many real-life applications. We cannot give more details of the extensions due to space constraints.

Post Processor As explained in the previous sections, the component level recognition results can be combined to get the word level output. At this stage, one could use a post processor to improve the recognition accuracy. We use two distinct types of post processing schemes. In an application for a learning aid for the illiterate people, we have a fixed and limited word set to be recognized. We use post-processing based on geometric information of components for some confusing pairs. At this stage, some of the frequent misclassifications are also corrected using approximate string matching techniques. A generic post processor for Indian script needs to use the language specific details for recognition. A morphological analyzer, developed in-house [2] and a spell-checker based on this has also been integrated with the OCR system.

Document Speaking System Another application we have developed is a reading aid for the visually impaired people. With the scarcity of printed material in Braille, it is becoming more and more difficult for such people to access information in the printed medium. The document reading system integrates an in-house developed Text-To-Speech (TTS) system [2] with the OCR to achieve this. A data driven approach using example-based learning is the basis for the TTS. An integrated document reading system will be tested for field applicability in the next six months.

Document Database System An important application of OCR is for converting printed documents into electronic form on which search and retrieval can be carried out. Indian language document databases are nonexistent today primarily because of the lack of good OCR systems. We have integrated the OCR engine with a page segmentation procedure for the archival of printed pages. Wherever OCR fails to recognize the text, the relevant area is stored as an image. The rest of the text is converted into electronic form(ISCII) and the content and structure of the text block is stored. Many content and structure based queries are supported.

6. Conclusions and Future Work

This paper presents an SVM-PCA based OCR for two major Indian languages. Many complementary aspects like (a) Phonetic nature of the language and its reflection on the script (b) Similarity and Dissimilarity between Indian scripts based on their origin and evolution (c) Geometrical/shape features for characterization of characters are not addressed at this stage. We plan to extend this for Indian scripts like Malayalam, Tamil, Kannada, Bangla, Oriya in the next couple of years. The OCR engine will be distributed to promote applications based on this.

Acknowledgments

The authors wish to thank Dr. P. J. Narayanan, Mr. S. P. Kishore, Prof. Rajeev Sangal and the Language Technologies Research Centre, Hyderabad, for many suggestions and contributions.

References

[1] http://www.cedar.buffalo.edu/ilt/research.htm.

[2] http://www.iiit.net/ltrc/index.html.

[3] Indian language document analysis and understanding. *Special Issue of Sadhana*, 2002.

[4] S. Antanani and L. Agnihotri. Gujarati character recognition. *Proc. ICDAR 1999*, pages 418–421, 1999.

[5] V. Bansal and R. M. K. Sinha. A devanagari OCR and a brief review of OCR research for Indian scripts. In *Proc of STRANS01*, 2001.

[6] B. B. Chaudhuri and U. Pal. An OCR system to read two Indian language scripts: Bangla and devanagari (hindi). *in Proc of ICDAR*, pages 1011–1015, 1997.

[7] C. V. Jawahar, M. N. S. S. K. Pavan Kumar, and S. S. Ravi Kiran. A bilingual OCR for hindi-telugu documents. *Technical Report TR-CVIT-22, IIIT,Hyderabad*, 2002.

[8] John.C.Platt, N. Cristianini, and J. Shawe-Taylor. Large margin dags for multi-class classification. In *Advances in Neural Information Processing Systems 12*, pages 547–553, 2000.

[9] T. Kanungo et al. A statistical, nonparametric methodology for document degradation model validation. *IEEE Tran. PAMI*, pages 1209–1223, 2000.

[10] M.Vidyasagar. *A Theory of Learning and Generalization*. Springer-Verlag,New York, 1997.

[11] G. Nagy. Twenty years of document image analysis in pattern analysis and machine intelligence. *IEEE Tran. PAMI*, pages 38–82, 2000.

[12] A. Negi, C. Bhagvathi, and B. Krishna. An OCR system for telugu. In *Proc. of ICDAR*, 2001.