Bigram language models and reevaluation strategy for improved recognition of online handwritten Tamil words

SURESH SUNDARAM, Indian Institute of Technology, Guwahati
A G RAMAKRISHNAN, Indian Institute of Science

The present article describes a post processing strategy for online handwritten isolated Tamil words. Contributions have been made with regard to two issues, hardly addressed in the online Indic word recognition literature, namely use of (1) language models exploiting the idiosyncrasies of Indic scripts and (2) expert classifiers for the disambiguation of confused symbols.

The input word is first segmented into its individual symbols, which are recognized using a primary Support Vector Machine (SVM) classifier. Thereafter, we enhance the recognition accuracy by utilizing (i) bigram language model at the symbol or character level and (ii) expert classifiers for reevaluating and disambiguating the different sets of confused symbols. The symbol level bigram model is used in a traditional Viterbi framework. The concept of a character comprising multiple symbols is unique to Dravidian languages such as Tamil. This multi-symbol feature of Tamil characters has been exploited in proposing a novel, prefix-tree based character level bigram model that does not use Viterbi search; rather it reduces the search space for each input symbol, based on its left context.

For disambiguating confused symbols, a dynamic time warping approach is proposed to automatically identify the parts of the online trace that discriminates between the confused classes. Fine classification of these regions by dedicated expert SVMs reduces the extent of confusions between such symbols. The integration of segmentation, prefix-tree based language model and disambiguation of confused symbols is presented on a set of 15,000 handwritten isolated online Tamil words. Our results show recognition accuracies of 93.0% and 81.6% at the symbol and word level, respectively, as compared to the baseline classifier performance of 88.4% and 65.1%, respectively.

Categories and Subject Descriptors: I.5.4.f [**Handwriting Analysis**]: Pattern Recognition; I.7.5.d [**Optical Character Recognition**]: Document and Text processing; I.5.2.c [**Pattern Analysis**]: Pattern recognition

General Terms: Experimentation, Language, Performance

Additional Key Words and Phrases: Online Tamil words, Support Vector Machines (SVM), Reevaluation, Language Models, Expert Classifiers.

1. INTRODUCTION

Tamil is a Dravidian language spoken predominantly by a significant population in the southern region of India. The language is written using the 'Tamil script' and is written from left to right. In this article, we address various practical aspects related to recognizing online handwritten isolated Tamil words. As is evident from the recent literature, there has been very little work exploring the problem of online word

This work is supported by the TDIL, Department of Information Technology, Govt of India.
Author's addresses: Medical Intelligence and Language Engineering Laboratory, Department of Electrical Engineering, Indian Institute of Science, Bangalore, Karnataka, India.
Department of Electrical Engineering, Indian Institute of Technology, Guwahati , India. Email: sureshsundaram@iitg.ernet.in, ramkiag@ee.iisc.ernet.in

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

recognition in Indian languages. The earliest work on online Tamil character recognition has been that of [Sundaresan and Keerthi 1999]. They evaluated the performance of angle, Fourier and wavelet features on a neural network classifier. Amongst these features, they show that wavelet features are quite effective as they retain both the intra-class similarity and inter-class differences. A combination of time-domain and frequency-domain features has been attempted with a Hidden Markov Model (HMM) classifier by [Toselli et al. 2007]. A similar set of feature combinations has been recently tested with an elastic matching approach in [Prasanth et al. 2007]. For writer dependent online handwriting recognition of isolated Tamil characters, a comparative study of elastic matching schemes using Dynamic Time Warping (DTW) has been presented in [Joshi et al. 2004]. Three different features are considered namely, preprocessed x-y co-ordinates, quantized slope values and dominant point co-ordinates. However, the writer dependent set up, in a way, limits the experimental validation. A subspace based classification approach has been proposed by [Deepu et al. 2004]. Principal component analysis (PCA) is applied separately to feature vectors extracted from the training samples of each class. The subspace formed by the first few eigenvectors is considered to represent the model for that class. During recognition, the test sample is projected onto each subspace and the class corresponding to the one that is closest is declared as the recognition result. A similar methodology of class-specific subspace classification has been adopted in [Sundaram and Ramakrishnan 2008] [Sundaram and Ramakrishnan 2009] using the two dimensional PCA (2DPCA) technique.

Different strategies for prototype selection for recognizing handwritten characters of Tamil script are investigated in [Raghavendra et al. 2005]. In particular, for modeling the differences in the complexity of different character classes, a prototype set growing algorithm is proposed with DTW+nearest neighbor as the classifier. A method of prototype learning is discussed in [Niels and Vuurpijl 2005] to speed up the recognition with the DTW framework. The authors in [Swethalakshmi et al. 2007] propose a set of offline-like features to capture information about both the positional and structural (shape) characteristics of the handwritten unit. The obtained features are then fed to a Support Vector Machine (SVM) for classifying the pattern. In [Aparna et al. 2004], unique strokes in the script are manually identified and each stroke is represented as a string of shape features. The test stroke is compared with the database of such strings using the proposed flexible string matching algorithm. The sequence of stroke labels is recognized as a character using a finite state automaton (FSA). Rule based approaches, as described in [Swethalakshmi et al. 2007] and [Aparna et al. 2004], are too script specific and may not generalize well to different writing styles. Reference [Kiran et al. 2010] provides a comparative study of statistical DTW and HMM on Tamil symbols.

Despite the different classification methodologies being addressed for the problem of isolated online Tamil character recognition, confusion between symbols does arise, that affect the performance of the handwriting recognition. This is primarily due to the fact that the techniques rely on decisions made by a single classifier, that operate on features at a global level. Though it has been acknowledged in state of the art literature that the confusions between online handwritten Tamil symbols arise primarily as a result of high degree of structural similarity, no attempts have been made so far to specifically come up with strategies of resolving the same. This is an important area to look at, considering the fact that the problem of writer independent recognition is quite complex to be solved by a single classifier alone [Vuurpijl et al. 2003].

In contrast to isolated online Tamil symbol recognition, there are very few works in the literature [Bharath and Madhvanath 2007] [Bharath and Madhvanath 2012] dedicated to the recognition of online Tamil words. In [Bharath and Madhvanath 2007], each symbol is modeled using a left-to-right HMM. Inter-symbol pen-up strokes were modeled explicitly using two-state left-to-right HMMs to capture the relative positions between symbols in the word context. Independently built symbol models and inter-symbol pen-up stroke models were concatenated to form the word models. The approach is segmentation-free and is tested with lexicons of varying sizes. An extension to this work is reported in [Bharath and Madhvanath 2012]. Here, a Bag-of-Symbols (BoS) representation of the handwritten word is proposed to address the issue of symbol order variations within and across characters.

Both the lexicon-driven and lexicon-free approaches proposed in [Bharath and Madhvanath 2012] to recognize the dataset containing only 85 distinct Tamil words and 70 distinct Hindi words, are limited to using a lexicon for improving the accuracy. However, for real-life applications, one may encounter out of vocabulary words. Hence, it is important to come up with an approach that can well generalize to such scenarios.

2. CONTRIBUTIONS OF THE PRESENT WORK

The main contribution of the article lies in integrating a post-processing module, with the aim of providing a complete solution to the recognition problem (Figure 1). The idea is to enhance the recognition of online Tamil words beyond that provided by the primary classifier. The post-processing techniques proposed in this work, namely incorporation of language models and disambiguation of confused symbols have not been adequately explored in the literature for the recognition of online Indic scripts.

The use of bigram language models has been proposed in two independent and distinct ways. The symbol level bigram model has been applied in the traditional Viterbi lattice to obtain the output sequence of labels with the best joint posterior probability. On the other hand, the bigram model at the character level is employed without the use of Viterbi selection; whenever a symbol is recognized, the character bigram statistics is used to restrict the search space for the next symbol to a subset of the total number of recognition classes.

Researchers in the recent literature attribute a part of their recognition errors to the presence of symbols that appear visually similar, which confuse the classifier. The classifiers usually work on features at the global level, and so, at times, fail to capture the subtle differences that distinguish between these symbols. One way to address this issue is to incorporate a framework that employs class-specific features and a bank of expert classifiers to improve the recognition of frequently confused characters. However, to come up with such features, one needs to identify the parts of the trace that can well differentiate between two or more similar looking characters. Accordingly, we propose a novel technique to learn the fine nuances of the confused characters using a dynamic time warping scheme. Each expert works on the discriminative part of the trace thus learnt to improve the recognition of the handwriting system. Thus, if the label assigned to an input pattern (after the language model) belongs to one of the frequently confused symbols, it is reevaluated by the appropriate set of expert classifiers. In other words, the experts aim to correct wrong decisions, if any, from the language model block. The resulting postprocessing framework (comprising the integration of one of the two distinct language models at a time with disambiguation of confused symbols by experts) improves recognition results at the word-level.

3. TAMIL SCRIPT AND THE CHOICE OF SYMBOL SET

The original Tamil script comprises 12 pure vowels, 18 pure consonants and a special character /ah/. The pure consonants get modified by each of the 12 vowels to generate a total of $18 \times 12 = 216$ consonant-vowel (CV) combinations. These add up to a total of 247 Tamil characters. In this work, however, we have included five additional pure consonants (used to represent the consonants borrowed from Sanskrit) [Aparna and

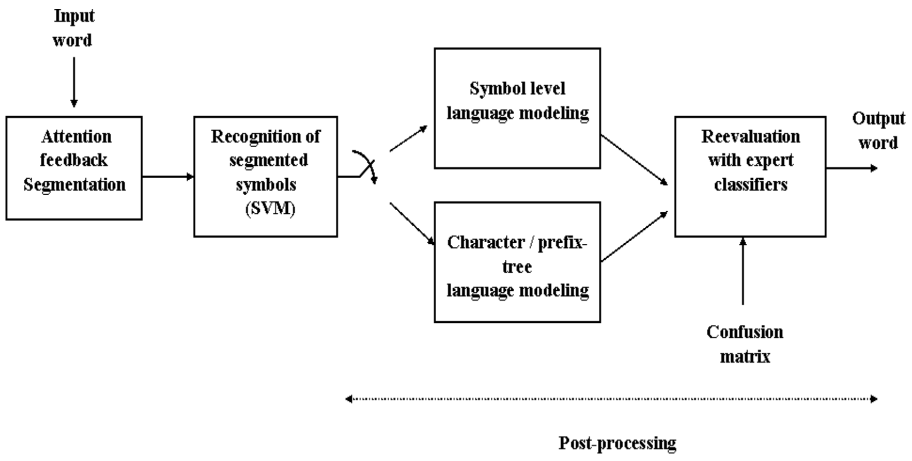

Input word
Attention feedback Segmentation
Recognition of segmented symbols (SVM)
Symbol level language modeling
Character / prefix-tree language modeling
Reevaluation with expert classifiers
Output word
Confusion matrix
Post-processing

Fig. 1. Block diagram of our complete recognition system for online handwritten Tamil words. The segmentation module is a two step process (termed 'Attention feedback segmentation') and is discussed in [Sundaram and Ramakrishnan 2013].

Ramakrishnan 2002] and another special symbol /sri/. These consonants contribute an additional $5 \times 12 = 60$ CV combinations. The complete 313 character set consists of 276 CV combinations, 12 vowels, 23 pure consonants and two special characters [Nethravathi et al. 2010]. All of these characters are supported by Unicode [1].

Analysis of the complete character set indicates that the characters may appear in one of the forms listed below. Based on these observations, we come up with a strategy to choose the minimum number of entities/ symbols for recognizing the 313 characters, taking into account the fact that many of the symbols may be written with a single or multiple strokes.

(1) Pure consonants modified by the inherent vowel அ /a/ are referred to as 'base consonants'. A few examples of base consonants are க /ka/, ச /ca/ and ள /La/. These base consonants give rise to 23 distinct classes that can be considered for recognition.
(2) The vowel modifier for ஆ /A/, written as ா appears to the right of the base consonant in the CV combination. Examples are கா /kA/, தா /tA/ and யா /yA/. Thus, these 23 CV combinations add only the symbol ா as a new class for recognition.
(3) In the CV combinations of vowels இ /i/ and ஈ /I/, the vowel modifier overlaps with the base consonant. Examples of such CVs are கி /ki/, சீ /cI/, ழி /zhi/ and ளீ /LI/. We consider these $23 \times 2 = 46$ CV combinations as additional distinct classes for recognition, since sometimes they are written using a single stroke.
(4) When the vowel உ /u/ or ஊ /U/ combines with any one of 18 of the 23 pure consonants, the basic shapes of the base consonants are altered. A few illustrations of these CV combinations are பு /pu/, ழு /zhu/, கு /ku/ and சூ /cU/. In the case of the CV combinations of the other five consonants (namely ஸ /sa/ , ஷ /sha/, ஜ /ja/, ஹ /ha/ and க்ஷ /ksha/), the shape of the base consonant is unaltered with the vowel modifier overlapping with it on the top. Examples of such CV combinations are ஸு /su/, க்ஷு /kshu/, ஸூ /sU/ and ஹூ /hU/. These CV combinations thus add another $23 \times 2 = 46$ classes for recognition.

[1] http://www.unicode.org/charts/PDF/U0B80.pdf

(5) In the CV combinations of vowels எ /e/, ஏ /E/ and ஐ /ai/, the corresponding vowel modifiers ெ, ே and ை spatially appear as a distinct/separate entity to the left of the base consonant being modified. Examples of such CV combinations are நெ /ne/, யே /yE/ and கை /kai/. Thus, these 46 CV combinations add only the vowel modifiers ெ, ே and ை as new classes for recognition.
(6) CV combinations of vowels ஒ /o/, ஓ /O/ and ஔ /au/ comprise two distinct entities with the base consonant sandwiched between them. The characters பொ /po/, டோ /TO/ and கௌ /kau/ illustrate such CV combinations. Since all these 4 symbols have already been added in the context of base consonants and CV combinations discussed above in (1), (2) and (5), there is no additional class contributed by these CV combinations.
(7) The vowel ஔ /au/ comprises 2 distinct entities- ஒ /o/ and ள /La/ that have already been considered as a vowel and base consonant, respectively. Hence, there is no necessity to represent it as a separate class for recognition.

With the above analysis, it is found that the set of 155 distinct classes (henceforth referred to in this work as ‘symbols’) is sufficient to form (and hence recognize) all the 313 characters considered. The 155 distinct symbols comprise:

(1) 11 pure vowels (excluding ஔ /au/)
(2) 23 pure consonants
(3) 23 base consonants
(4) 23 CV combinations of இ /i/
(5) 23 CV combinations of ஈ /I/
(6) 23 CV combinations of உ /u/
(7) 23 CV combinations of ஊ /U/
(8) 6 Additional symbols (ா (VM of /A/) , ெ (VM of /e/) , ே (VM of /E/), ை (VM of /ai/), ஃ /ah/ and ஸ்ரீ /sri/.)

It is to be noted that a Tamil character may consist of at most 3 symbols. We refer to characters comprising more than one symbol as ‘multi-symbol’ in this work.

4. DESCRIPTION OF THE DATASETS USED FOR THE STUDY

In this study, even though we deal with the recognition of handwritten word data collected by us, the primary as well as the expert classifiers have been trained using isolated character data from an external database.

4.1. Details of the training dataset

Based on the symbol set described in the earlier section, Hewlett Packard Labs (HP Labs) India in the International Workshop on Frontiers in Handwriting Recognition (IWFHR) 2006 released a corpus comprising isolated online Tamil symbols for research [Madhvanath and Lucas 2006]. We refer to this dataset as ‘IWFHR Tamil symbol set’ in this work. We used the training set of this corpus for training our primary classifier. This classifier is then applied to the test set of this corpus (comprising 26926 samples) to construct the confusion matrix and to pick the sets of frequently confused symbols for disambiguation. Thereafter, we revert to the training samples of each confused set to learn the discriminative regions of their traces and to build their expert classifiers. Further details on the expert classifiers are systematically described in subsections 7.1, 7.2, 7.3 and 7.4 of the article.

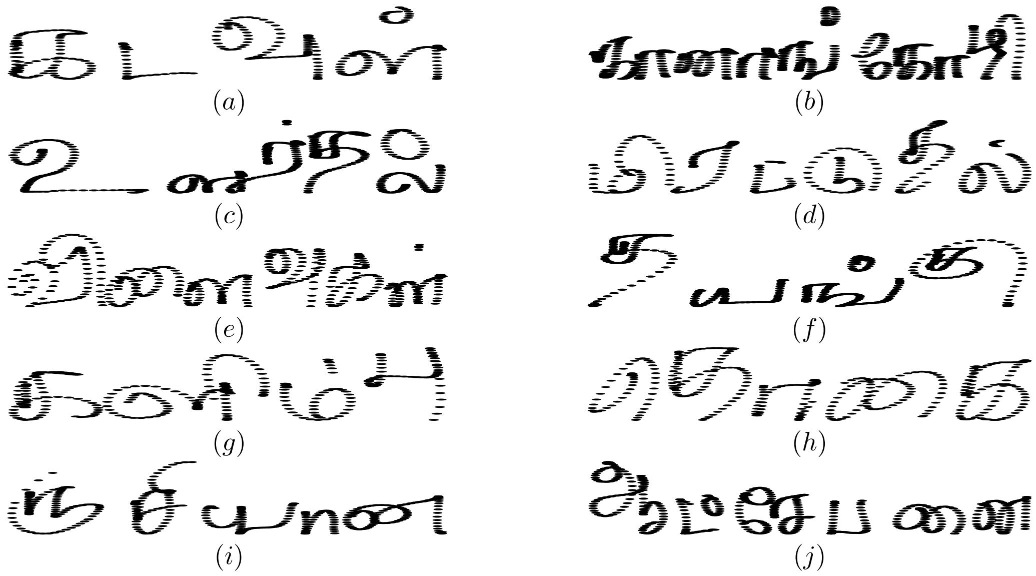


Fig. 2. Few samples of online acquired handwritten Tamil words from the word database.

4.2. Details of the test word dataset

The present work is an outcome of an ongoing funded project from the Technology Development for Indian Languages (TDIL) program of the Ministry of Information Technology of the Government of India. The main agenda of the project is to create a writer independent recognition framework suitable for form filling applications such as encumbrance certification and census data collection. Since every such form involves proper name and address fields, the requirement is also to look for open vocabulary recognition. Isolated Tamil words have been collected using a custom application running on a tablet PC. We have ensured that all the writers, who participated in the data collection activity, are native Tamil speakers, who currently write in that language, at least irregularly. Accordingly, we came across different popular writing styles for Tamil symbols. Moreover, the participants were provided with a graphics interface with large rectangular boxes and were prompted to write Tamil words with minimal constraints , one in each box in the form. No restrictions were placed on the number of strokes, shape of the symbols and direction of the constituent strokes. Figures 2 (a)-(j) present a few sample words from our database. This database is available for research purposes from our lab server [2]

High school and college students from many educational institutions in the Indian state of Tamil Nadu contributed in building the word data-base of size 15,000 comprising 2,000 distinct words. The set of words has been chosen to cover all the 155 symbols of the Tamil script. The collected word samples are stored using the UNIPEN v1.0 format [Guyon et al. 1994]. The collected words consist of 80,098 labelled Tamil symbols. We use the labelled symbols as the ground truth for computing the recognition accuracies. The 15,000 word data contain several instances of all the 2,000 word classes. Each word class has 7 to 8 instances in the test set. The word set excludes symbols written with delayed, overwritten and extraneous strokes and cursively written words. The bar distribution representing the frequency of each the 155 symbols in the word dataset is shown in Table. I

[2]http://mile.ee.iisc.ernet.in/OHRDataset/

Table I. Occurrence statistics of different groups of the 155 Tamil symbols in the word database.

Type of symbols	% of total # of occurrences
Pure vowels	4.3
Base consonants	36.4
CV combinations of ி /i/	9.5
CV combinations of ீ /I/	0.8
CV combinations of ு /u/	9.4
CV combinations of ூ /U/	0.8
Pure consonants	22.5
Additional symbols	16.3

We re-emphasize that (i) the training data for the primary and the expert classifiers and (ii) the information on confusion sets, are all derived from IWFHR Tamil symbol set, which is distinct from the word dataset that we have collected. The latter (word) data is only used as test data.

5. PRIMARY SVM CLASSIFIER

In this article, we extend the work presented in [Sundaram and Ramakrishnan 2013] and present a full-fledged system to recognize online handwritten, isolated Tamil words. We first segment an online Tamil word to its constituent symbols in a two-step process. The resulting segments are recognized by the primary classifier to generate valid symbol labels, which, in turn, are concatenated to generate the output word. Owing to the good generalization capability of the SVM classifier to unseen test data, we adopt it as the primary classifier in this work.

Let the input word W be segmented to p symbols, $\{S_i\}_{i=1}^{p}$. Each of these symbols is a sequence of x-y coordinates with pen-up and pen-down events. A pre-processing step, applied prior to recognition, compensates for variations in time, scale and velocity. It comprises 3 steps : (1) smoothing (2) normalization (3) resampling [Joshi et al. 2004; Deepu et al. 2004]. The final result of pre-processing is a new sequence of points $\{x_i, y_i\}_{i=1}^{n}$ regularly spaced in arc length. A feature vector $\mathbf{x}$ is constructed from this sequence and fed to the SVM classifier.

$$\mathbf{x} = (x_1, x_2....x_n, y_1, y_2, ...y_n) \tag{1}$$

We experimented with varying number of resampled points and observed that $n_P = 60$ is quite sufficient in capturing the shape of the character, including points of high curvature. The vector $\mathbf{x}$ is referred to as the 'concatenated x-y coordinates' in this work.

The performance of the SVM classifier is largely dependent on the selection of the parameters. The training data of the IWFHR symbol set are employed to obtain the model parameters. We have employed the LIB-SVM software [Chang and Lin 2011] with the Radial Basis Function (RBF) chosen as the kernel. The type of kernel and the corresponding parameters have been set optimally after performing five-fold cross validation experiments .

The primary SVM classifier, fed with concatenated x-y coordinates of the preprocessed segmented symbols, is found to be quite effective for the task of segmentation [Sundaram and Ramakrishnan 2013]. However, the classifier is not robust to effectively distinguish between similar looking symbols. With the view of improving the performance of symbol recognition beyond that given by the primary classifier, we propose in the remainder of this article, two post-processing approaches, namely bigram language models and confused-character disambiguation.

Two distinct, independent bigram models have been proposed in this work. These have been described in Sections 6 and 8, respectively. The reevaluation techniques, aimed at correcting the decisions made by language model, are discussed in Section 7.

Table II. Multi-symbol Consonant Vowel (CV) combinations illustrating the differences between the symbol order and the order of the Unicode representations. 'BC' in the second column is an abbreviation to 'Base Consonant'. The third column shows the order in which symbols are written in the CV combinations of a BC /ya/. The last column presents the order of the Unicodes for those combinations.

Vowel forming the CV	Symbol writing order	Example of CV character	Unicode order
ஆ /A/	BC+ா	யா /yA/	BC+ா
எ /e/	ெ +BC	யெ /ye/	BC+ெ
ஏ /E/	ே + BC	யே /yE/	BC+ே
ஐ /ai/	ை +BC	யை /yai/	BC+ை
ஒ /o/	ெ + BC+ா	யொ /yo/	BC+ெ....ா
ஓ /O/	ே + BC+ா	யோ /yO/	BC+ே....ா
ஔ /au/	ெ + BC+ள	யௌ /yau/	BC+ெ....ள

The performance of the proposed post processing schemes on a set of 15,000 words is described in Section 9. Section 10 summarizes the article.

6. LANGUAGE MODELS

The goal of a language model is to exploit the linguistic regularities and characteristics in the recognition framework. There are quite a few works in the current literature on the use of language models for recognition of handwriting in non-Indic scripts ([Vinciarelli et al. 2004] [Quiniou and Anquetil 2006] [Quiniou et al. 2005] [Perraud et al. 2003] [Li and Tan 2004b] [Li and Tan 2004a] [Marti and Bunke 2000] [Zimmermann and Bunke 2004]). Contrary to that, in the area of online recognition of Indic scripts, there is hardly any work incorporating the use of language models [Bharath and Madhvanath 2009].

An extensive Unicode text corpus, comprising 1.5 million Tamil words is utilized for generating the frequency count of each of the 155 symbols. The text has been derived from the Project Madurai site [3] and the EMILLE Beta Version text corpus [4]. The corpus essentially is a collection of sentences, wherein each word comprises a sequence of Tamil characters. The Unicode corpus was first transformed to a corpus of Tamil symbols, by inverse mapping from the Unicode sequence of Tamil characters to the corresponding symbol sequences. This is essential, since the symbol order and the order of the Unicodes are different for different sets of CV combinations as shown in Table II. We consider the statistics of the symbols obtained from this corpus to be representative of the script. Moreover, a multi-symbol character may be composed of as many as three symbols. From the corpus, we derive the following statistics.

— N_T - Total number of occurrences of all the symbols.

— $N_s(\omega_i)$ - Total number of occurrences of symbol ω_i.

— $N_{ss}(\omega_i, \omega_j)$ - Number of occurrences of the symbol pair (ω_i, ω_j). Any one or both of them can be single symbol characters or vowel modifiers.

— $N_{cs}(c_i, \omega_j)$ - Number of occurrences of symbol ω_j following the multi-symbol character c_i. The symbol ω_j can be a single symbol character or the first part of a multi-symbol character.

[3]http://www.projectmadurai.org/
[4]http://www.lancaster.ac.uk/fass/projects/corpus/emille/

Table III. Illustrative examples for the joint occurrence of various symbols and/or multi-symbol characters. The occurrences of such pairs in the text corpus are counted to generate the linguistic statistics.

Nature of pair	Examples
Pair of symbols (ω_i, ω_j)	(ச, மு) (ப, தி) (ெ, ந) (ை, த) (/ca/, /mu/) (/pa/, /ti/) (VM of /e/, /na/) (VM of /ai/,/ta/)
Symbol and multi-symbol character (ω_i, c_j)	(ச, கை) (ப, யா) (ப, யோ) (அ, கா) (/ca/, /kai/) (/pa/, /yA/) (/pa/, /yO/) (/a/, /kA/)
Multi-symbol character followed by a symbol (c_i, ω_j)	(கை, ள) (யா, ரு) (யோ, க) (கா, டி) (/kai/, /La/) (/yA/, /ru/) (/yO/, /ka/) (/kA/, /Ti/)
Pair of multi-symbol characters (c_i, c_j)	(கை, யா) (நெ, யோ) (யோ, டோ) (கா, பொ) (/kai/, /yA/) (/ne/, /yO/) (/yO/, /TO/) (/kA/, /po/)

— $N_{sc}(\omega_i, c_j)$ - Number of occurrences of the multi-symbol character c_j following symbol ω_i, which can be a single symbol character or the final part of a multi-symbol character.

— $N_{cc}(c_i, c_j)$ - Number of occurrences of the multi-symbol character pair (c_i, c_j).

Table III presents illustrations for each of the above pairs.

A specific word W can be interpreted as a realization of a discrete stochastic process. Two different models are proposed to probabilistically describe the interdependencies of symbols in W namely (1) n-gram language models and (2) n-class models. As such, an n-gram/ n-class model is interpreted as a Markovian model of order n-1. In the present work, we consider n=2, namely the bigram and biclass models.

6.1. Proposed Symbol level bigram model

Given an online Tamil word W, recognized as $\{\omega_i\}_{i=1}^{p}$, we can write its probability (assuming a full order Markov process) as

$$\begin{aligned} P(W) &= P(\omega_1, \omega_2.....\omega_p) \\ &= P(\omega_1)P(\omega_2|\omega_1)P(\omega_3|\omega_1,\omega_2)....P(\omega_p|\omega_1,\omega_2...\omega_{p-1}) \end{aligned} \tag{2}$$

However, it becomes very unrealistic and demanding to obtain statistics for higher order Markovian processes. In our work, we have considered only the first order Markovian dependency.

The simplest language model called the 'unigram model' treats the symbols of a word to be independent of each other. However, the actual probability of occurrence of a symbol, as determined from the corpus, is accounted for. Using this model, we can write

$$P(W) = P(\omega_1)P(\omega_2).....P(\omega_p) \tag{3}$$

where

$$P(\omega_i) = \frac{N_s(\omega_i)}{N_T} \tag{4}$$

In the bigram model, the probability of occurrence of a symbol in a word depends only on the immediately preceding symbol. This model incorporates a first order Markovian

Table IV. Occurrence statistics of different groups of the 155 Tamil symbols, as derived from the text corpus.

Group	Type of symbols	Total # of occurrences	% of total # of occurrences
G_1	Base consonants	368387	33.5144
G_2	Pure consonants	266525	24.2474
G_3	Additional symbols	191282	17.4021
G_4	CV combinations of உ /u/	104360	9.4943
G_5	CV combinations of இ /i/	99421	9.0449
G_6	Pure vowels	57858	5.2637
G_7	CV combinations of ஈ /I/	6252	0.5688
G_8	CV combinations of ஊ /U/	5105	0.4644

dependency and accordingly we can rewrite the probability of the word as

$$P(W) = P(\omega_1)P(\omega_2|\omega_1)...P(\omega_i|\omega_{i-1})...P(\omega_p|\omega_{p-1}) \tag{5}$$

where

$$P(\omega_i|\omega_{i-1}) = \frac{N_{ss}(\omega_{i-1}, \omega_i)}{N_s(\omega_{i-1})} \tag{6}$$

It is quite possible for a symbol or pair of symbols in the word to be recognized to have never occurred in the corpus [Marti and Bunke 2000]. In order to incorporate a non-zero probability to the bigram statistics for such symbols, we smooth the language model. The idea is to reduce the probabilities of bigrams occurring in the corpus and redistribute this mass of probabilities among bigrams never encountered. One simple smoothing technique is to pretend each bigram occurs once more than it actually does. This is accomplished by the following updation.

$$P(\omega_j|\omega_i) = \frac{1 + N_{ss}(\omega_i, \omega_j)}{155 + N_s(\omega_i)} \tag{7}$$

6.2. Proposed symbol level biclass model

Biclass models merge the symbols into groups [Perraud et al. 2003]. In this section, we use the words ‘group’ and ‘class’ interchangeably. In order to form meaningful classes, we club symbols that are linguistically similar and create the eight groups ($G_1 - G_8$), as listed in Table IV. It can be observed that the groups occur with varying frequencies in the text corpus.

We consider the first order Markovian dependency between the classes (groups), wherein a Tamil symbol is assigned to exactly one group. Dedicated SVM classifiers are designed to compute the probability estimate of the symbol placed in a specific group. Accordingly, one can write for a bi-class model,

$$P(\omega_i|\omega_{i-1}) = P(\omega_i|G^{\omega_i}, \mathbf{x}^{S_i})P(G^{\omega_i}|G^{\omega_{i-1}}) \tag{8}$$

G^{ω_i} refers to the group to which the recognized symbol ω_i belongs. The first term $P(\omega_i/G^{\omega_i}, \mathbf{x})$ corresponds to the probability estimate (returned by the SVM classifier) for the symbol ω_i to belong to group G^{ω_i}. The second term is the prior probability of the group G^{ω_i} to occur after $G^{\omega_{i-1}}$ and can be readily derived from the corpus. One advantage of bi-class models is their compactness in representation. Because symbols are combined in groups, the number of bi-class probabilities is lower than that of bi-grams.

6.3. Word recognition using symbol level models

Let X represent the sequence of feature vectors for an online handwritten word, consisting of p symbol patterns $\{S_i\}_{i=1}^{p}$. The aim of word recognition is to find the most

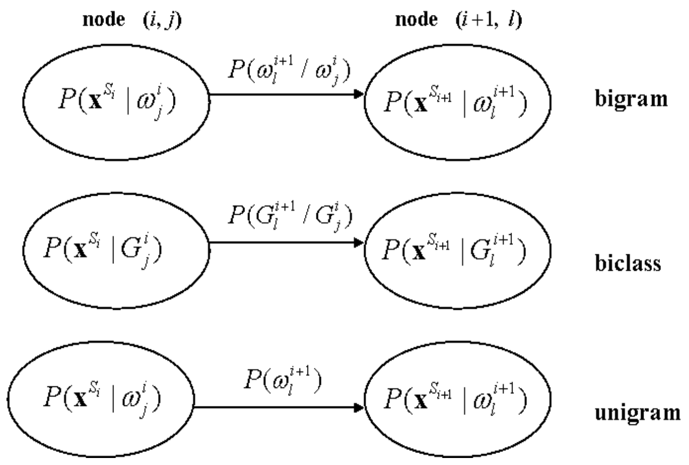

node (i, j)
node $(i+1, l)$
$P(\omega_l^{i+1} / \omega_j^i)$
$P(\mathbf{x}^{S_i} | \omega_j^i)$
$P(\mathbf{x}^{S_{i+1}} | \omega_l^{i+1})$
bigram
$P(G_l^{i+1} / G_j^i)$
$P(\mathbf{x}^{S_i} | G_j^i)$
$P(\mathbf{x}^{S_{i+1}} | G_l^{i+1})$
biclass
$P(\omega_l^{i+1})$
$P(\mathbf{x}^{S_i} | \omega_j^i)$
$P(\mathbf{x}^{S_{i+1}} | \omega_l^{i+1})$
unigram

Fig. 3. Illustration of a pair of nodes in a word graph. The nodes represent the probability estimates of the symbol returned from the SVM classifier. The links denote the contextual dependency between adjacent symbols (as captured in bigrams, biclass and unigram models).

plausible sequence of symbols $\hat{W}$ for X.

$$\hat{W} = \arg\max_{W} p(W|X) \tag{9}$$

W represents the set of candidate symbol sequences for X. From Bayes rule, we can write

$$\hat{W} = \arg\max_{W} \frac{p(X|W)p(W)}{p(X)} \tag{10}$$

Here $p(X|W)$ represents the probability estimate of the handwritten word for the given candidate sequence W. $p(W)$ is the prior probability of W derived from the language model. The denominator $p(X)$ is independent of W and hence is ignored. Thus,

$$\hat{W} = \arg\max_{W} p(X|W)p(W) \tag{11}$$

We use the decimal logarithmic representation for the various probabilities and write

$$\hat{W} = \arg\max_{W} \; [\log_{10}(p(X|W)) + \log_{10}(p(W))] \tag{12}$$

The optimal sequence of symbols for the handwritten word can be traced using the well known Viterbi algorithm [Rabiner and Juang 1986]. Assuming context-free, independent shape recognition for each pattern S_i by the classifier, we can write

$$p(X|W) = \Pi_{i=1}^{p} p(\mathbf{x}^{S_i}|\omega_i) \tag{13}$$

The unigram (Eqn 3) and the bigram models (Eqn 5) are used to provide the estimates for $P(W)$.

We now describe the structure of the word recognition system. The preprocessed x-y coordinates (feature vector $\mathbf{x}$) of every symbol of the segmented word is input to the SVM classifier, which outputs a list of M (chosen as four in this work) candidate symbols ordered by their probability estimates. A word graph is then created with these choices. In that graph, $(i, j)^{th}$ node represents the probability estimate $P(\mathbf{x}^{S_i}|\omega_j^i)$ of the j^{th} recognized symbol for i^{th} segment S_i. In the case of bigram models, the edges between the nodes (i, j) to $(i + 1, l)$ represents $P(\omega_l^{i+1}|\omega_j^i)$. For unigrams, the edges are given by the prior probability $P(\omega_l^{i+1})$ in the corpus. Let G_j^i represent the group assigned for the j^{th} recognized symbol for i^{th} segment. Then, for the case of biclass models, we denote the edge link by $P(G_l^{i+1}|G_j^i)$. Figure 3 presents a pictorial representation of a pair of nodes of a word graph for the three different models.

7. COMBINATION OF REEVALUATION WITH LANGUAGE MODELS

In this section, we address an issue, that does at times, lead to an erroneous symbol with the bigram model. Let the optimal symbol sequence of the word $\hat{W}$ from the bigram model be defined as

$$\hat{W} = \{\hat{\omega_i}\}_{i=1}^{p} \tag{14}$$

We consider the actual symbol sequence of the online Tamil word W as

$$W = \{\omega_i\}_{i=1}^{p} \tag{15}$$

If the word $\hat{W}$ differs from W in exactly one position, say j , then the bigram language model favors $\hat{\omega}_j$ to ω_j whenever

$$\begin{aligned} & & \hat{\omega}_i = \omega_i \;\; i \neq j \\ p(\mathbf{x}^{S_j}|\hat{\omega}_j)P(\hat{\omega}_j|\hat{\omega}_{j-1}) & > & p(\mathbf{x}^{S_j}|\omega_j)P(\omega_j|\hat{\omega}_{j-1}) \end{aligned} \tag{16}$$

In such cases, total dependence only on the bi-gram language model unduly favors one of the two confused symbols, given the same context. We need to rectify the symbol $\hat{\omega}_j$ to ω_j. One can consider resolving the confusion by extracting a set of discriminative features from regions of the trace in the symbols $\hat{\omega}_j$ and ω_j, that differ structurally. In other ways, we reevaluate the label of $\mathbf{x}^{S_j}$.

We illustrate here one such situation where reevaluation is necessitated, since language models cannot, by themselves, deliver. In Tamil, a verb can be modified by forms of tense, number, gender and person. Each verb results in a new word after each of these morphological changes. Considering verbs modified with gender, the ones associated with masculine gender end with the symbol ன் /n/, while those with feminine gender end with ள் /L/. Examples of such words include (வந்தான் (he came), வந்தாள் (she came)) and (வருகிறான் (he comes) , வருகிறாள் (she comes)). Note that the words in each pair differ only by the symbols ன் /n/ and ள் /L/ at the last position. Interestingly, the symbols ன் /n/ and ள் /L/ get confused with each other by the SVM classifier. All the remaining symbols of the word being the same, from Eqn. 16, the bigram model favors the more likely symbol of the confusion set (ன், ள்) at the last position. Thus, at times, the wrong symbol may be preferred to the correct one, leading to an error. Therefore, strategies are invoked to disambiguate (ன், ள்) to output the right symbol.

We now propose a methodology to reduce the confusions between such symbols.

7.1. Identification and resolution of confusions

In order to find the possible confusions, we run the SVM classifier across the 26936 test samples of the IWFHR Tamil symbol set. A few of the similar looking pairs with their frequencies of confusion and their recognition accuracies from the primary SVM classifier are listed in Table V .

Let $\mathbf{C}$ represent the confusion matrix of size 155×155. The term $c_{i,j}$ in $\mathbf{C}$ corresponds to the number of samples of symbol ω_i getting wrongly classified as ω_j.

Table V. Some symbol confusions encountered at the output of the primary classifier (SVM) and their frequency of occurrence in the IWFHR 2006 Tamil test symbol set.

Symbol pairs	Total # of symbols in the test set	# of confusions output from SVM	Primary classifier recognition accuracy in %
(மு ,ழு) (/mu/, /zhu/)	349	26	92.6
(ண ,ை) (/Na/, VM of /ai/)	351	32	90.9
(ளி ,ணி) (/Li/, /Ni/)	364	32	91.2
(ள ,ண) (/La/, /Na/)	353	23	93.5
(கி ,சி) (/ki/, /ci/)	355	17	95.2
(ல ,வ) (/la/, /va/)	359	14	96.1

$$\mathbf{C} = \begin{pmatrix} c_{1,1} & c_{1,2} & \cdots & \cdots & c_{1,155} \\ c_{2,1} & & \cdots & \cdots & c_{2,155} \\ \cdot\cdot & & & & \\ \cdot\cdot & & & & \\ c_{155,1} & & \cdots & \cdots & c_{155,155} \end{pmatrix}$$

Accordingly, for a symbol pair (ω_i, ω_j) , we can write the number of confusions as

$$c_T(i,j) = c_{i,j} + c_{j,i} \tag{17}$$

For a given symbol ω_i, the set of symbols to which it gets frequently confused is given by

$$\Omega_i = \{\omega_j | c_T(i,j) \geq \delta, i \neq j\}. \tag{18}$$

We choose the threshold δ, so that only confusion sets having a frequency above 3% in the confusion matrix are considered for disambiguation. We denote the set of all symbols that possibly get confused as

$$\Omega = \bigcup_i \Omega_i \tag{19}$$

Visual inspection of the confused symbols indicates that they appear different in some critical parts of the trace, while retaining certain common structures. For example, the confused symbols ல `/la/` and வ `/va/` differ mainly in the middle portion of the online trace. The confusion pair க `/ka/` and சு `/cu/` appear structurally different towards the end of the trace.

In this work, we address the disambiguation of confusion pairs by proposing a set of dedicated classifiers referred to as 'experts'. Figure 4 presents the block diagram of an expert. We design independent expert networks for each confusion set. Each expert consists of three blocks: (a) discriminative region extractor (b) feature extractor and (c) SVM classifier. Let (s_1, s_2) denote a confusion pair, as obtained from the confusion matrix $\mathbf{C}$. Then for each confusion (s_1, s_2), the corresponding expert extracts the specific discriminative region $\Re(s_1, s_2)$ from the input symbol pattern. This region corresponds to those parts of the trace that capture the finer structural nuances in s_1 and s_2. The

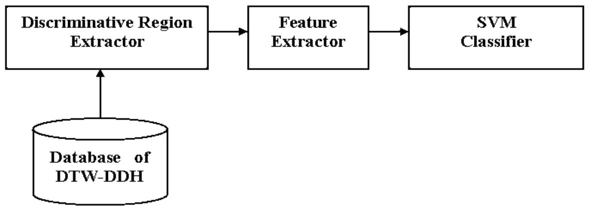


Fig. 4. Component blocks of an expert used for disambiguating a confusion pair.

feature extractor extracts features from $\Re(s_1, s_2)$, which in turn are fed to the SVM classifier for disambiguation.

7.2. Extraction of discriminative region for a confusion pair

In this subsection, we propose a novel methodology to automatically locate the discriminative parts of strokes in confused pairs. In the domain of off-line handwriting recognition, extraction of the distinctive image regions in confused patterns have been addressed in the works [Rahman and Fairhurst 1997; Leung and Leung 2010; Xu et al. 2010]. However, in the context of online recognition of Indic scripts, such attempts have not been made. In this article, we exploit the available temporal information of the trace for learning the finer parts that distinguish the confused symbols. Accordingly, let (s_1, s_2) represent a confusion symbol pair. We use the training patterns of s_1 and s_2 in the IWFHR Tamil symbol set to learn the discriminative parts of the online trace $\Re(s_1, s_2)$. Let $N_{Tr}^{s_1}$ and $N_{Tr}^{s_2}$ correspond to the number of training data for s_1 and s_2, respectively. Each such pattern is preprocessed (using the steps presented in Section 5) and described by $\{(x_1, y_1), (x_2, y_2), ..., (x_{n_P}, y_{n_P})\}$ We use the DTW approach (described in [Joshi et al. 2004]) to learn the structural differences between the traces of the confused pairs s_1 and s_2. Essentially, we elastically match each temporal sequence in the training set $N_{Tr}^{s_1}$ against each of the sequences in $N_{Tr}^{s_2}$. This in turn gives rise to $N_{Tr}^{s_1} \times N_{Tr}^{s_2}$ different DTW cost matrices with corresponding optimal warping paths.

Let $(x_k^{s_1}, y_k^{s_1})$ and $(x_l^{s_2}, y_l^{s_2})$ respectively denote the k^{th} and l^{th} points in the confusion pair s_1 and s_2. The $(k, l)^{th}$ element in the cost matrix describes the degree of dissimilarity $d(k, l)$ between these points.

$$d(k, l) = \sqrt{(x_k^{s_1} - x_l^{s_2})^2 + (y_k^{s_1} - y_l^{s_2})^2} \tag{20}$$

The optimal path $\mathcal{W}^*$ in each cost matrix has some sections with low values of $d(k, l)$ corresponding to similar regions in the confused pair of symbols and other section or sections with high values corresponding to the discriminative parts. We utilize this property to obtain the discriminative region $\Re(s_1, s_2)$.

We generate a histogram to accumulate the pen positions of the trace contributing to the structural differences in (s_1, s_2). This histogram (referred to as the 'DTW discriminative distance histogram' (DTW-DDH)) has n bins. For a DTW match between a pair of training patterns from s_1 and s_2, we first record the maximum dissimilarity cost obtained in the optimal path $\mathcal{W}^*$. Thereafter, we compare the remaining costs along $\mathcal{W}^*$ to this value. The indices (in $\mathcal{W}^*$) that have costs greater than a threshold T_d are voted in the DTW-DDH. This procedure is repeated for each of the $N_{Tr}^{s_1} \times N_{Tr}^{s_2}$ DTW cost matrices. The net result is an accumulation of votes for the sample indices. Subsequent to the construction of the DTW-DDH, we analyze the peaks. The peaks in the histogram correspond to the repeated occurrence of higher costs in the differ-

ALGORITHM 1: Pseudo-code for generating the DTW-DDH for a given confusion pair (s_1, s_2)

```
Let $(s_1, s_2)$ be a confused symbol pair.
$N_{Tr}^{s_1}$ = No. of training samples of $s_1$ in the IWFHR Tamil symbol set.
$N_{Tr}^{s_2}$ = No. of training samples of $s_2$ in the IWFHR Tamil symbol set.
Initialize the DTW-DDH by setting the votes for each of the $n_P$ sample indices to zero.
for $i = 1 : N_{Tr}^{s_1}$
    for $j = 1 : N_{Tr}^{s_2}$
        Elastically match the temporal sequence of $i^{th}$ training pattern of $s_1$ to the $j^{th}$ training pattern of $s_2$ by using the DTW algorithm.
        Retrace the optimal DTW path $\mathcal{W}^*$.
        Record the maximum dissimilarity $d_{max}^{i,j}$ cost along $\mathcal{W}^*$.
        Increment the votes for those sample indices of the trace in $\mathcal{W}^*$, whose dissimilarity measure exceeds a threshold $T_d$. The threshold $T_d$ is adapted to the maximum dissimilarity value $d_{max}^{i,j}$ between the $i^{th}$ training pattern of $s_1$ and the $j^{th}$ training pattern of $s_2$ .
    End
End
```

ent warping paths $\mathcal{W}^*$ and denote the possible regions for discriminating the symbols (s_1, s_2). We present the pseudocode for construction of the DTW-DDH in Algorithm 1.

We set T_d to 90% of the maximum dissimilarity cost along a given warping path $\mathcal{W}^*$. This value is sufficient for identifying the region of finer nuances that discriminate the confusion pairs (s_1, s_2). The threshold is adaptive and varies with the maximum dissimilarity cost obtained in each of the $N_{Tr}^{s_1} \times N_{Tr}^{s_2}$ warping paths. Figure 5 illustrates the DTW-DDH obtained from the training samples of the confusion set (ல `/la/`, வ `/va/`). The sample index corresponding to the bin having the largest number of votes in the DTW-DDH gives rise to the maximum peak. Around this peak, a window of samples is considered to describe the part of trace distinguishing the confusion pair s_1 and s_2. This, in turn, forms the discriminative region (DR) $\Re(s_1, s_2)$.

It is to be noted that, owing to different styles of writing, different transients occur at the start and/or end of the online trace. This in turn creates peaks at the start and/or end of the DTW-DDH. Such peaks correspond to the votes of the first and last sample indices in the DTW-DDH and are not included for analysis. From the DTW-DDH of the symbols ல `/la/` and வ `/va/`, we observe that the main peak occurs in the middle region, thereby indicating that the discriminative region lies in the middle part of the trace.

For a given confusion pair (s_1, s_2), we now address the issue of selecting the window size for extracting the discriminative region $\Re(s_1, s_2)$. We adapt the window to a given DTW-DDH of a confusion pair (s_1, s_2). The size of the window is automatically decided by the set of sample indices around the main peak that have votes above 30%.

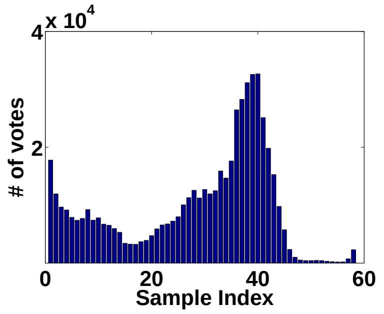


Fig. 5. DTW discriminative distance histogram (DTW-DDH) corresponding to the symbols /la/ and /va/ obtained using their samples from IWFHR Tamil symbol training set.

7.3. Features for confusion disambiguation

From the above discussion, it is evident that each confusion pair (s_1, s_2) has a discriminative region $\Re(s_1, s_2)$, that can be obtained from its DTW-DDH. The concatenated x-y coordinates in the region $\Re(s_1, s_2)$ are used as the features for disambiguating the confused pairs s_1 and s_2.

Let the size of the window used for extracting the $\Re(s_1, s_2)$ in the DTW-DDH be denoted by L. One can then represent the part of trace captured in $\Re(s_1, s_2)$ by $\{(x_b, y_b), (x_{b+1}, y_{b+1}), ..., (x_{b+L-1}, y_{b+L-1})\}$. A feature vector of dimension $2 \times L$ is constructed from this sequence and fed to the expert SVM classifier for disambiguation.

$$\mathbf{x}^{(s_1,s_2)} = (x_b, x_{b+1}, x_{b+2}....x_{b+L-1}, y_b, y_{b+1}, ...y_{b+L-1}) \tag{21}$$

Here, (x_b, y_b) is the first sample index of the DR for the given confusion pair.

7.4. Classification strategy of Expert SVMs

Several experts can exist for a given symbol ω_i. Hence, in order to disambiguate a pattern recognized as symbol ω_i, it is passed through a set (or a bank) of expert classifiers. In fact, the number of experts in the bank corresponds to the number of symbols in Ω_i (defined in Section 7.1) that can be possibly confused with ω_i. Each expert for a confusion pair extracts the features from its discriminative region and passes it to a trained SVM classifier. The recognition scores from each of the expert SVMs, together with their labels are noted. Subsequently, based on the best recognition probability estimate obtained, we make a decision to choose its corresponding symbol as the reevaluated output for the pattern.

For a confusion pair (s_1, s_2), we use their corresponding training patterns from the IWFHR Tamil symbol set to learn the expert SVM. The features are derived from the discriminative parts of the online trace $\Re(s_1, s_2)$ (as described in the previous subsection). The RBF kernel is used in our experimentation. We have employed the LIB-SVM software [Chang and Lin 2011] for learning the SVM model parameters. The kernel and the corresponding parameters are optimally set after performing five-fold cross validation experiments on the training patterns.

8. PREFIX-TREE BASED CHARACTER LEVEL LANGUAGE MODELS

As explained in Section 3, the CV combinations of the vowels ஆ /A/, எ /e/, ஏ /E/ and ஐ /ai/ are made up of two distinct symbols and those of ஒ /o/, ஓ /O/ and ஔ /au/ are written with three distinct symbols. Thus, a multi-symbol Tamil character can comprise up to three distinct symbols. We consider the symbols in a Tamil word to be drawn from the finite vocabulary $\mathbf{V} = \{\omega_k\}_{k=1}^{155}$. It is assumed that the input word has been segmented into its constituent symbols [Sundaram and Ramakrishnan 2013] [Sundaram and Ramakrishnan 2010].

In this section, we propose ways in which context information aids in reducing the number of symbol classes to be tested for an input pattern. We describe the contextual information in terms of the positional and bigram statistics derived at the character level. A bigram at the level of characters is equivalent to a higher order language model, where the context deciding the occurrence probability of a symbol can involve up to five previous symbols, thus essentially constituting a 6-gram model at the symbol level.

It is important to clarify the distinction between the use of the bigram model at character-level (proposed in this section) and that at the symbol level (described in section 6). The bigram language model at the symbol level is applied post recognition to choose the maximum probability path (word) among the multiple, possible paths in the Viterbi trellis. On the other hand, the language model at the character level is applied prior to recognition to eliminate some of the symbols from being considered as probable for the segmented symbol pattern. The set of possible labels for a symbol depends on its contextual history, (described in terms of the recognized labels of past symbols that form a Tamil character). Here, at every position of the word, it is assumed that the recognition up to the previous symbol (position) is correct. Accordingly, the current input symbol is recognized against only a subset of the 155 symbols. By considering the history in terms of the character, the probability of all the symbols that have zero bigram probability are removed from the search space for the test pattern under consideration. Owing to the fact that a Tamil character may comprise up to 3 symbols, the search space is adaptive based on the recognized history of each symbol. In contrast to word recognition using the symbol-level bigram models, the bigram model at the character level does not rely on the optimal Viterbi path for obtaining the output word. The proposed character-level bigram approach is quite analogous to the prefix tree concept in the field of computer science. Hence the name -‘Prefix-tree based language models’.

— Let us look at the first symbol position of a word. Now, language constraints specify that a word cannot begin with certain symbols. This can be obtained from the zero values of the probability $P(\omega_1)$.
Let $\mathbf{F}_0$ represent the set of symbols that never occur at the starting position of a word. There are 68 such symbols that make up the set $\mathbf{F}_0$. For a pattern S_1, occurring at the first position in W, we can reduce the search space roughly by a factor of two by excluding the symbols in $\mathbf{F}_0$ for recognition. We denote the subset of symbols, serving as likely candidates for the segmented pattern at the start of a word, by $\mathbf{L}_1$. Accordingly, we can write

$$\mathbf{L}_1 = \mathbf{V} \setminus \mathbf{F}_0 \tag{22}$$

where $\setminus$ denotes the set difference operator.

— As we move to symbol positions beyond the first, the complete past context is always kept track of. The search space for any symbol S_i at an arbitrary position $i, (1 < i < p)$ in a word gets limited by its immediate past (prefix), that can possibly contain up to 5 symbols.
For the symbol S_i, let $\{\omega_{i-k}\}_{k=1}^{i-1}$ denote the set of recognized symbols that precede it. We present below the various distinct contexts possible for any symbol and the precluded symbols for each such context:

(1) If the recognition label ω_{i-1} corresponds to a complete Tamil character, then

$$\mathbf{F}_1 = \{\omega_j | N_{ss}(\omega_{i-1}, \omega_j) = 0\} \tag{23}$$

(2) If
  (a) ω_{i-1} corresponds to the initial part of a 2 symbol CV combination and
  (b) ω_{i-2} is a single symbol character or space token $< s >$
then,

$$\mathbf{F}_2 = \{\omega_j | N_{sc}(\omega_{i-2}, cv_1) = 0\} \tag{24}$$

Here cv_1 is generated using the 2 symbols ω_{i-1} and ω_j. Accordingly, based on the contextual history of 2 preceding recognition labels $\omega_{i-1}, \omega_{i-2}$, the set $\mathbf{F}_2$ represents the symbols that never occur at the i^{th} position of the word.

(3) If $\{\omega_{i-1}, \omega_{i-2}\}$ correspond to a character cv_2 , then

$$\mathbf{F}_3 = \{\omega_j | N_{cs}(cv_2, \omega_j) = 0\} \tag{25}$$

(4) If $\{\omega_{i-1}, \omega_{i-2}, \omega_{i-3}\}$ correspond to a character cv_3 ,

$$\mathbf{F}_4 = \{\omega_j | N_{cs}(cv_3, \omega_j) = 0\} \tag{26}$$

On the basis of the context history of 3 preceding recognition labels, $\mathbf{F}_4$ represents the set of symbols that never occur at the i^{th} position of the word.

(5) If
  (a) ω_{i-1} corresponds to the leading part of a 2 symbol CV combination and
  (b) recognition labels ω_{i-3}, ω_{i-2} form a 2 symbol character cv_4 ,
then,

$$\mathbf{F}_5 = \{\omega_j | N_{cc}(cv_4, cv_5) = 0\} \tag{27}$$

cv_5 is a character formed by ω_{i-1} and ω_j. $\mathbf{F}_5$ represents the set of symbols that never occur at the i^{th} position of the word, based on, the contextual history of 3 symbols $\{\omega_{i-3}, \omega_{i-2}, \omega_{i-1}\}$.

(6) If
  (a) ω_{i-1} corresponds to the first part of a 2 symbol CV combination and
  (b) recognition labels ω_{i-4}, ω_{i-3}, ω_{i-2} form a 3 symbol character cv_6,

$$\mathbf{F}_6 = \{\omega_j | N_{cc}(cv_6, cv_7) = 0\} \tag{28}$$

cv_7 is a character formed by ω_{i-1} and ω_j. $\mathbf{F}_6$ is the set of symbols that never occur at the i^{th} position of the word, given the contextual history of 4 recognized labels $\omega_{i-4}, \omega_{i-3}, \omega_{i-2}, \omega_{i-1}$.

(7) If
  (a) $\omega_{i-2}, \omega_{i-1}$ correspond to the initial part of a 3-symbol character and
  (b) ω_{i-5}, ω_{i-4}, ω_{i-3} form a 3 symbol character cv_8,

$$\mathbf{F}_7 = \{\omega_j | N_{cc}(cv_8, cv_9) = 0\} \tag{29}$$

cv_9 is a 3 symbol character formed by the symbols $\omega_{i-2}, \omega_{i-1}, \omega_j$. $\mathbf{F}_7$ represents the set of symbols that never occur at the i^{th} position of the word, when the contextual history corresponds to the 5 symbols ω_{i-5}, ω_{i-4}, $\omega_{i-3}, \omega_{i-2}, \omega_{i-1}$.

— For a pattern S_p, occurring at the end of a word, we can further reduce the search space by excluding the symbols in $\mathbf{F}_8$ for recognition. Here $\mathbf{F}_8$ represents the set of symbols that never occur at the end of a word.

A symbol under consideration will only belong to one of the above contexts and accordingly, the reduced search space described by the set $\mathbf{L}_i$, is used to recognize it. Thus if j^{th} contextual information is applicable for the i^{th} symbol, then one can write,

$$\mathbf{L}_i = \mathbf{V} \setminus \mathbf{F}_j \quad j \in 1, 2, ..7 \tag{30}$$

8.1. An illustration of the prefix-tree language model in reducing the search space for recognition

We now illustrate the application of the prefix-tree language model for the recognition of a Tamil word யோகம் /yOkam/ (refer Table VI) in a step-by-step manner using the prefix-tree.

— The pattern at the start of the word is tested against the 87 symbols in $\mathbf{L}_1$ with the SVM classifier and the most probable symbol ே (VM of vowel /E/) is assigned to it.

— To recognize the second pattern, we use its contextual information of the previous recognized symbol ே. The symbol ே is a vowel modifier of ஏ /E/. In fact, it corresponds to the the initial part of a 2 symbol CV combination of /E/. In order to form a complete character (from context 2), we constrain the current pattern to be recognized to the set of 15 base consonants that can follow ே. Accordingly, the SVM returns the symbol ய /ya/ as the most probable for this pattern.

— We constrain the third pattern to be recognized only against those symbols that can follow the 2 symbol character யே (context 3). From the set of 16 symbols, the SVM returns ா as the most probable symbol for this pattern. Though it is not a complete character, we make use of the prior knowledge that this symbol always follows a base consonant and associate it to the previous character யே to form another valid character யோ /yO/ (consonant ய modified by the vowel ஓ).

— To recognize the fourth pattern, we rely on the contextual prior information of its preceding character யோ (context 4). Thus, we constrain the pattern to be recognized only against the 15 symbols that can follow this 3 symbol character. Accordingly, the SVM returns symbol க /ka/ as the most probable label for this pattern. The recognized symbol க is a character by itself.

— The prior context for the last pattern is the single symbol க. Accordingly, we constrain this pattern to be recognized against the subset of 76 symbols and obtain ம் /m/ as the most probable label.

For comparison, Table VII illustrates the reduction in the search space for each symbol pattern by considering only the previous symbol label. It is interesting to observe that the number of symbols to be tested is higher than that obtained using the context of the previous character label (see Table VI).

9. RESULTS AND DISCUSSION

In this section, we evaluate the performance of the proposed post processing system, comprising language modeling and confused-character disambiguation (reevaluation). For ease of understanding the experimental results, we hereinafter refer to the recog-

Table VI. Application of the prefix tree character-level language model in reducing the search space for the recognition of symbols segmented from a handwritten Tamil word . For each input pattern, we show the reduced number of symbols to be tested against, (in the third column), based on its context of past recognized character (given in the second column). The number of symbols being precluded from recognition is dependent on the context history. These are presented in the last column.

Input symbol pattern	Contextual information	# of symbols to be tested	# of symbols being precluded from recognition
1	S_b	L_1 =87	F_0 = 68
2	ே (VM of /E/)	L_2 =15	F_2 = 140
3	யே /yE/	L_3 =16	F_3 = 139
4	யோ /yO/	L_4 =15	F_4 = 140
5	க /ka/	L_5 = 76	F_1 = 79

Table VII. Illustration of the possible reduction in the search space for the input symbol pattern by considering only the previous symbol label (use of symbol level bigram statistics). Note that the number of classes (symbols) to be tested is higher than that obtained using the context of the previous character label (character level bigram statistics).

Input symbol pattern	Contextual information	# of symbols to be tested
1	S_b	87
2	ே (VM of /E/)	15
3	ய /ya/	80
4	ா (VM of /A/)	110
5	க /ka/	76

nition framework described in [Sundaram and Ramakrishnan 2013] (without the post processing module) as the **baseline system**.

9.1. Word recognition performance using the proposed symbol-level language models

In order to set up the bi-class language model (described in Sec 6.2), a SVM classifier is separately trained, specific to the symbols in each of the groups $G_1 - G_8$. Table VIII presents the details of the designed classifiers with their recognition performance on the IWFHR Tamil symbol test set.

In the experiments outlined in this subsection, we compare the performance of the n-gram and class-based language models, with and without the reevaluation module. In order to incorporate the influence of linguistic knowledge, we weighted the second term of Eqn 12 by a factor β (ranging between 0 and 1) as presented below.

$$\hat{W} = \arg\max_{W} \; [\log_{10}(p(X|W)) + \beta \log_{10}(p(W))] \tag{31}$$

$\beta = 0$ corresponds to baseline system, while $\beta = 1$ provides an equal weighting to both the recognition and the language model. Figure 6 presents the symbol recognition rate for values of β from 0 to 1 in steps of 0.1 on a validation set of 250 words. The three

Table VIII. Group-wise symbol recognition performances of the SVM classifiers on the test samples of IWFHR Tamil symbol dataset. Each classifier was trained on the specific group of symbols $(G_1 - G_8)$ from the training set.

Classifier	Group	Recognition accuracy (in %)
C_b	G_1	95.6
C_p	G_2	93.5
C_o	G_3	98.8
C_u	G_4	91.2
C_i	G_5	95.6
C_v	G_6	97.3
C_I	G_7	95.7
C_U	G_8	89.7

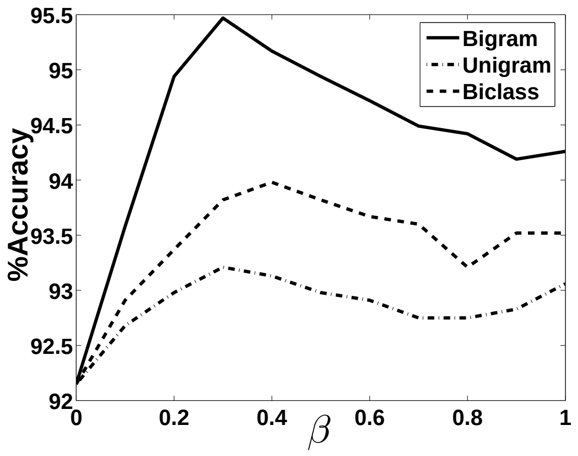


Fig. 6. Variation of symbol recognition accuracy obtained for different choices of the weights β used for the language models. The experiments are conducted on a validation set of 250 words.

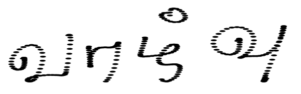
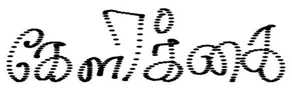
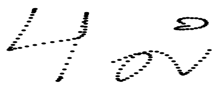
curves (corresponding to unigram, biclass and bigram language models) show their behavior and the optimal value of β is 0.3 for the unigram and the bigram models and 0.4 for the biclass model. On an average, irrespective of β, the bigram model outperforms the unigram model by 2%. Furthermore, we can see the importance of this weight since the symbol recognition rate is 94.3 % with the bigram model when $\beta = 1$ whereas it is 95.5 % with the optimal value of β. One can also observe that the biclass model performs lower than that of the bigram model, but better than the baseline system and unigram model. An improvement of up to 2% is achieved with respect to the baseline system.

The symbol recognition accuracies for each model is obtained across the word database (Table IX). We notice that the bigram model outperforms the others in terms of recognition performance. Table X shows a few sample words that have been corrected by imposing the bigram language model on the baseline SVM recognition system. The wrongly recognized symbols are highlighted by square boxes in the third column. From Table IX, across the 80,098 symbols in the word database, we notice an improvement of 3.7% (from 88.4% to 92.1%) and 1.3% (from 88.4% to 89.7%) in symbol recognition performance over the baseline classifier for the bigram and unigram models. The incorporation of the bigram models gives a word recognition accuracy of

Table IX. Performance comparison of the different n-gram models with Viterbi selection on the recognition of symbols in the word database, with and without reevaluation. Number of words tested = 15000. Number of symbols = 80098.

	Symbol recognition accuracy (in %)	Word recognition accuracy (in %)
Baseline system	88.4	65.1
Unigram model	89.7	68.0
Bigram model	92.1	77.6
Bi-class model	90.3	72.4
Unigram+reevaluation	90.8	73.2
Bigram+reevaluation	92.9	80.8
Biclass model+reevaluation	91.3	74.8

Table X. Examples of words that have been wrongly recognized by the baseline SVM classifier but corrected by the application of the symbol-level bigram model in a Viterbi trellis .

Sl.No	Input handwritten word	Output of baseline classifier	Word recognized using bigram model
1		வரழ்வு /varazhvu/	வாழ்வு /vAzhvu/
2		கேலிக்கை /kElikkai/	கேளிக்கை /kELikkai/
3		புஸீ /pusI/	புல் /pul/

77.6%.

Table XI lists a few sample words that have not been corrected by the bigram language model (refer column 3). As discussed in Sec 7, the symbol errors occur due to the optimal path chosen by the Viterbi encoding scheme, which heavily depends on the bias in the bigram statistics between adjacent symbols. For such scenarios, reevaluation strategies are invoked on the output symbols returned by the optimal Viterbi path for possible corrections (shown in column 4). For all the four words, the reevaluation of base consonants is employed for correcting the erroneous symbols. From Table IX, incorporation of the reevaluation strategies on the output from the bigram language model enhances the symbol recognition from 92.1% to 92.9%. In summary, a judicious combination of reevaluation strategies with a language model improves the symbol recognition performance beyond that provided by the language model alone. The combination of bigram model with reevaluation enhances the word recognition rate of the baseline classifier by 15.7% (from 65.1% to 80.8%)

9.2. Enhancement of word recognition performance with character level based prefix-tree language model

In this subsection, we evaluate the performance of the prefix-tree based language models on the word database with and without the reevaluation module. The incorporation of the different contexts for reducing the search space for the test pattern shows an improvement of 1.7% (from 88.4% to 90.1%) in symbol recognition accuracy

Table XI. Examples of words that have been wrongly recognized by the baseline SVM classifier and bigram model in Viterbi selection, but corrected after reevaluation.

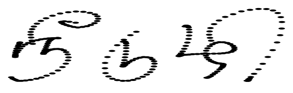
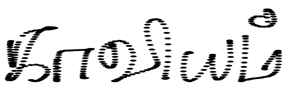
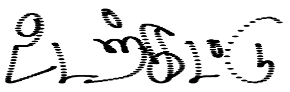
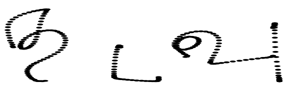
Sl.No	Input handwritten word	Word recognized using bigram model (without reevaluation)	Word recognized using bigram with reevaluation
1		நீடுமி /NIdumi/	நீடுழி /NIduzhi/
2		காவியப் /kAviyap/	காவியம் /kAviyam/
3		உடர்கட்டு /uTarkaTTu/	உடற்கட்டு /uTaRkATTu/
4		நடவு /naTavu/	தடவு /taTavu/

Table XII. Performance evaluation of the prefix tree based character level language models on the recognition of symbols in the word database.

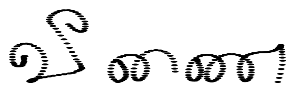
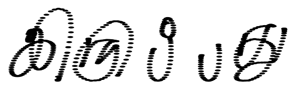
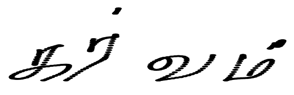
	Symbol recognition accuracy (in %)	Word recognition accuracy (in %)
Baseline system	88.4	65.1
Prefix tree based model	90.1	71.5
Prefix tree based model+reevaluation	93.0	81.6

over the baseline recognition system (Table XII). Correspondingly, the word accuracy improves by 6.5%.

A drawback of incorporating the prefix-tree language model alone is the possible propagation of symbol errors as depicted in the third column of Table XIII. This is attributed to the fact that such models make use of the contextual information provided by the immediately preceding character for recognition. Unlike symbol-level language models (described in Sec 6.3), we have not incorporated dynamic programming approaches like the Viterbi algorithm to obtain the optimal word. However the error propagation can be minimized to a great extent by revaluating the label of the current symbol by reevaluation strategies before proceeding to the next symbol (see fourth column of Table XIII). The combination of language models with reevaluation improves the symbol recognition rate by 4.6% (from 88.4% to 93.0%) and word recognition by 16.5% over the baseline system.

It is interesting to note that the recognition performances of symbol-level and prefix-tree based bigram models on the word database (92.9% and 93.0%) are comparable after the reevaluation strategies. With respect to the baseline classifier, the tree based bigram model with reevaluation enhances the word recognition rate by 16.5% (from 65.1% to 81.6%), as compared to 15.7% obtained by the bigram Viterbi approach.

Table XIII. Examples of words wrongly recognized by the reduced search deployment of the prefix-tree character-level bigram language models. Without reevaluation, propagation of error may occur, as observed from the words shown in the third column. The fourth column shows that the errors are corrected with reevaluation.

Sl.No	Input handwritten word	Word recognized using only prefix-tree model (without reevaluation)	Word recognized using prefix-tree model with reevaluation
1		வீணனி /vINaNi/	வீணை /vInnai/
2		இருபீமது /irupImatu/	இருப்பது /iruppatu/
3		கற்றும் /kaRRum/	கர்வம் /karvam/

9.3. Perplexity measure

One of the metrics for evaluating a language model is its perplexity [Marti and Bunke 2000]. For a test set W_T composed of t words $(W_1, W_2, ..., W_t)$ we can estimate the probability of $p(W_T)$ as the product of the probabilities of all the words in the set.

$$p(W_T) = \prod_{i=1}^{t} P(W_i) \tag{32}$$

In particular, given a language model that assigns probability $p(W_T)$ to the sequence of t words, we can derive a compression algorithm that encodes the words W_T using $-\log_2 p(W_T)$ bits. Let N_t represent the total number of symbols in the t words. The entropy H and perplexity P of a language model can be defined as

$$H = \frac{-\log_2 p(W_T)}{N_t} \tag{33}$$

$$P = 2^H \tag{34}$$

For our work, we have $t = 15,000$, corresponding to the number of words in the database. Intuitively, perplexity is regarded as the average number of symbols from which the current symbol can be chosen. Table XIV presents the perplexity measures for the different language models.

Table XIV. Perplexity of the various language models-unigram, bigram, prefix-tree.

Recognition system configuration	Baseline	Unigram	Bigram	Prefix-tree
Perplexity	155	34	26	6.8

9.4. Computational Complexity

In this subsection, we discuss the computational complexities involved for the prefix-tree language model and the symbol-level bigram based recognition using Viterbi path.

— **Symbol level language model**: Given a word comprising p segments, the symbol-level language model based recognition by the Viterbi algorithm has a maximal time complexity of $O(|V|^2 p)$ [Huang et al. 2001]. Here, $|V|$ represents the number of symbols in the vocabulary. For our case, $|V|$=155.
— **Prefix tree language model**: The prefix tree language model analyzes the segmented symbol pattern based on its context in terms of the recognized labels of the past symbols. Let L_i correspond to the set of symbols to be compared against for the i^{th} symbol of the word. Then one needs to make $|L_i|$ (cardinality of the set) comparisons for recognizing that symbol. Accordingly, with the incorporation of contextual information to each segment in the word, we get the overall complexity of the system as $O(\sum_{i=1}^{p} |L_i|)$. By noting that L_i is a subset of 155 symbols, we can write $|L_i| < |V|$. This in turn implies that $\sum_{i=1}^{p} |L_i| < p|V|$. Making use of the fact that $p|V| < p|V|^2$, one can write $O(\sum_{i=1}^{p} |L_i|) < O(|V|^2 p)$. Thus, we see that the prefix tree language model is less computational than the symbol level language model.

Based on our current implementation in MATLAB, on an average, it takes about 3 seconds to recognize each symbol in a Tamil word.

9.5. Comparison to related works

Given that there is no prior work reported on explicitly segmenting and recognizing online Tamil words, it is difficult to compare our method to a benchmark. This being said, the results reported in [Bharath and Madhvanath 2007] [Bharath and Madhvanath 2012] adopt an implicit-segmentation approach to recognize a set of 85 distinct words using Hidden Markov Models. Moreover, a different set of symbols (distinct from the one used in this work) has been used for generating the models. The authors in [Bharath and Madhvanath 2007] report word recognition rates of 98 % to 92.2% with different lexicon sizes (1K to 20K words). Their recent work in [Bharath and Madhvanath 2012] suggest recognition rates of 91.8% by presenting a combination of HMM-Based lexicon-driven and lexicon-free word recognizers. Both the lexicon-driven and lexicon-free approaches are limited to using a lexicon for improving accuracy.

Our method on the other hand views explicit segmentation followed by post processing techniques as the recognition paradigm. It is to be noted that our segmentation/recognition approach is not aided by a lexicon and hence can be applied in real-life applications, where one may encounter out of vocabulary words.

The writer-independent and lexicon-free segmentation-recognition approach developed in this work for online handwritten Tamil word recognition is promising. The best performance of 93.0% (achieved at symbol level) is comparable to the highest reported accuracy in the IWFHR 2006 Tamil symbol Competition [Madhvanath and Lucas 2006]. However, the latter one is on a database of isolated symbols (IWFHR Tamil symbol test set), whereas our accuracy is on a database of 15,000 words and thus, a product of segmentation, recognition and postprocessing strategies.

10. CONCLUSION AND FUTURE WORK

This article presented a post processing scheme to enhance the performance of a segmentation-driven recognition system for online handwritten Tamil words. In particular, we proposed the (i) use of bigram language models and (ii) expert classifiers for reevaluating and disambiguating the confused symbols in a Tamil word. We showed that a judicious combination of bigrams with reevaluation is an effective post processing strategy. As an alternative, we propose a prefix-tree based approach using character level bigrams for reducing the search space of symbols during recognition.

Future avenues for research include exploration of a strategy for grouping Tamil symbols, prior to recognition, using possibly shape based features and confusion pair analysis. In addition, since the isolated word data have been collected keeping in mind the usability of the recognition system for form filling applications, it would be interesting to extend our recognition strategy for online handwritten paragraphs in the future.

The proposed technique is able to handle the limited skew present in the data without any specialized preprocessing technique. Nevertheless, it may be conducive to modify the segmentation strategy and test our approach on applications that present heavily skewed data. Currently, our approach does not effectively recognize the words comprising symbols written as a different temporal sequence rarely encountered in Tamil script. Cursive writing in Tamil is rare. Thus words in which two or more symbols are written by a single stroke get incorrectly segmented and hence wrongly recognized. In the future, a stochastic framework such as Hidden Markov Models with an incorporation of online and offline features needs to be investigated to alleviate these shortcomings. In addition, the incorporation of lexicon may be considered in our segmentation/ recognition framework in finite vocabulary problems.

Nevertheless, to the best of our knowledge, this article is one of the pioneering works directed at addressing the issues in developing an explicit segmentation based recognition system for online isolated Tamil words.

ACKNOWLEDGEMENTS
The authors thank Technology Development for Indian Languages (TDIL), Department of Information Technology, Government of India for funding this research. Thanks are also due to the schools and colleges, who helped us collect the online Tamil handwritten word data and to the project staff, who annotated the same.

REFERENCES

APARNA, K. G. AND RAMAKRISHNAN, A. G. 2002. A complete Tamil optical character recognition system. In *Proc. Workshop on Document Analysis Systems*. 53–57.

APARNA, K. H., SUBRAMANIAN, V., KASIRAJAN, M., PRAKASH, G. V., CHAKRAVARTHY, V. S., AND MADHVANATH, S. 2004. Online handwriting recognition for Tamil. In *Proc. Int'l Workshop. Frontiers in Handwriting Recognition*. 438–443.

BHARATH, A. AND MADHVANATH, S. 2007. Hidden Markov models for online handwritten Tamil word recognition. In *Proc. Int'l Conf. Document Analysis and Recognition*. 506–510.

BHARATH, A. AND MADHVANATH, S. 2009. Online handwriting recognition for Indic scripts. In *Guide to OCR for Indic scripts*. 209–234.

BHARATH, A. AND MADHVANATH, S. 2012. HMM-based lexicon-driven and lexicon-free word recognition for online handwritten Indic scripts. *IEEE Trans. Pattern Analysis and Machine Intelligence 34,* 4, 670–682.

CHANG, C. C. AND LIN, C. J. 2011. Lib-SVM : a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology 2*, 1–39.

DEEPU, V., MADHVANATH, S., AND RAMAKRISHNAN, A. G. 2004. Principal component analysis for online handwritten character recognition. In *Proc. Int'l Conf. Pattern Recognition*. 327–330.

GUYON, I., SCHOMAKER, L., PLAMONDON, R., LIBERMAN, M., AND JANET, S. 1994. Unipen project of on-line data exchange and recognizer benchmarks. In *Proc. Pattern Recognition*. 29–33.

HUANG, X., REDDY, R., AND ACERO, A. 2001. *Spoken Language Processing: A Guide to Theory, Algorithm and System Development*. Pearson Hall.

JOSHI, N., SITA, G., RAMAKRISHNAN, A. G., AND MADHAVANATH, S. 2004. Comparison of elastic matching algorithms for online Tamil handwritten character recognition. In *Proc. Int'l Workshop. Frontiers in Handwriting Recognition*. 444–449.

KIRAN, S., PRASAD, K. S., KUNWAR, R., AND RAMAKRISHNAN, A. G. 2010. Comparison of HMM and SDTW for Tamil handwritten character recognition. In *Proc. Signal Proc and Commn*. 1–4.

LEUNG, K. C. AND LEUNG, C. H. 2010. Recognition of handwritten Chinese characters by critical region analysis. *Pattern Recognition 43,* 3, 949–961.

LI, Y. X. AND TAN, C. L. 2004a. An empirical study of statistical language models for contextual post-processing of Chinese script recognition. In *Proc. Int'l Workshop Frontiers in Handwriting Recognition*. 257–262.

LI, Y. X. AND TAN, C. L. 2004b. Influence of language models and candidate set size on contextual post-processing of Chinese script recognition. In *Proc. Int'l Conf on Pattern Recognition*. 537–540.

MADHVANATH, S. AND LUCAS, S. M. 2006. IWFHR 2006 online Tamil handwritten character recognition competition. In *Proc. Int'l Conf. Frontiers in Handwriting Recognition*.

MARTI, U. V. AND BUNKE, H. 2000. Unconstrained handwriting recognition: Language models, Perplexity and System performance. In *Proc. Int'l Workshop Frontiers in Handwriting Recognition*. 463–468.

NETHRAVATHI, B., ARCHANA, C. P., SHASHIKIRAN, K., RAMAKRISHNAN, A. G., AND KUMAR, V. 2010. Creation of a huge annotated database for Tamil and Kannada OHR. In *Proc. Int'l Conf. Frontiers in Handwriting Recognition*. 415–420.

NIELS, R. AND VUURPIJL, L. 2005. Dynamic time warping applied to Tamil character recognition. In *Proc. Int'l Conf. Document Analysis and Recognition*. 730–734.

PERRAUD, F., VIARD-GAUDIN, C., MORIN, E., AND LALLICAN, P. M. 2003. N-gram and n-class models for on line handwriting recognition. In *Proc. Int'l Conf on Document Analysis and Recognition*. 1053–1059.

PRASANTH, L., BABU, J., SHARMA, R., RAO, P., AND DINESH, M. 2007. Elastic matching of online handwritten Tamil and Telugu scripts using local features. In *Proc. Int'l Conf. Document Analysis and Recognition*. 1028–1032.

QUINIOU, S. AND ANQUETIL, E. 2006. A priori and a posteriori integration and combination of language models in an on-line handwritten sentence recognition system. In *Proc. Int'l Workshop Frontiers in Handwriting Recognition*. 403–408.

QUINIOU, S., ANQUETIL, E., AND CARBONNEL, S. 2005. Statistical language models for on-line handwritten sentence recognition. In *Proc. Int'l Conf on Document Analysis and Recognition*. 516–520.

RABINER, L. AND JUANG, B. 1986. An introduction to hidden Markov models. *IEEE ASSP Magazine 3,* 1, 4–16.

RAGHAVENDRA, B. S., NARAYANAN, C. K., SITA, G., RAMAKRISHNAN, A. G., AND SRIGANESH, M. 2005. Prototype learning methods for online handwriting recognition. In *Proc. Int'l Conf. Document Analysis and Recognition*. 287–291.

RAHMAN, A. F. R. AND FAIRHURST, M. C. 1997. Selective partition algorithm for finding regions of maximum pairwise dissimilarity among statistical class models. *Pattern Recognition Letters 18,* 7, 605–611.

SUNDARAM, S. AND RAMAKRISHNAN, A. G. 2008. Two dimensional principal component analysis for online Tamil character recognition. In *Proc. Int'nl Conf. Frontiers in Handwriting Recognition*. 88–93.

SUNDARAM, S. AND RAMAKRISHNAN, A. G. 2009. An improved online Tamil character recognition engine using post-processing methods. In *Proc. Int'l Conf. Document Analysis and Recognition*. 1216–1220.

SUNDARAM, S. AND RAMAKRISHNAN, A. G. 2010. Attention feedback based robust segmentation of online handwritten words. In *Indian Patent Office Reference*. No: 03974/CHE.

SUNDARAM, S. AND RAMAKRISHNAN, A. G. 2013. Attention-feedback based robust segmentation of online handwritten isolated Tamil words. *ACM Transactions on Asian Language Information Processing 12,* 1, 1–25.

SUNDARESAN, C. S. AND KEERTHI, S. S. 1999. A study of representations for pen based handwriting recognition of Tamil characters. In *Proc. Int'l Conf. Document Analysis and Recognition*. 422–425.

SWETHALAKSHMI, H., SEKHAR, C. C., AND CHAKRAVARTHY, V. S. 2007. Spatiostructural features for recognition of online handwritten characters in Devanagari and Tamil scripts. In *Proc. Int'l Conf. Artificial neural networks*. 230–239.

TOSELLI, A. H., PASTOR, M., AND VIDAL, E. 2007. On-line handwriting recognition system for Tamil handwritten characters. In *Proc. Pattern Recognition and Image Analysis*. 370–377.

VINCIARELLI, A., BENGIO, S., AND BUNKE, H. 2004. Offline recognition of unconstrained handwritten texts using HMMs and statistical language models. *IEEE Trans. Pattern Analysis and Machine Intelligence 26,* 6, 709–720.

VUURPIJL, L., SCHOMAKER, L., AND ERP, M. V. 2003. Architectures for detecting and solving conflicts: Two-stage classification and support vector classifiers. *Int'l J. Document Analysis and Recognition 5,* 4, 213–223.

XU, B., HUANG, K., AND LIU, C.-L. 2010. Similar handwritten chinese characters recognition by critical region selection based on average symmetric uncertainty. In *Proc. International Conference on Frontiers in Handwriting Recognition*. 527–532.

ZIMMERMANN, M. AND BUNKE, H. 2004. Optimizing the integration of a statistical language model in HMM based offline handwritten text recognition. In *Proc. Int'l Conf on Pattern Recognition*. 203–208.