International Journal of Engineering Research-Online
A Peer Reviewed International Journal
Articles available online http://www.ijoer.in

Vol.3., Issue.5., 2015
(Sept.-Oct.)

RESEARCH ARTICLE

ISSN: 2321-7758

IDENTIFICATION OF LANGUAGES AND ENCODINGS IN ONLINE DOCUMENT A RESULT PAPER

ARCHITA[1], NIKITA SAGAR[2]

[1]M.Tech Student, [2]Head of the Department (CSE)

CSE Department, Indus Institute of Engineering and Sciences

ABSTRACT

Now-a-days the increase in popularity of portable computing devices such as PDAs and handheld computers, non keyboard based methods for data entry are receiving more attention in the research communities and commercial sector. The Pen-based and voice-based inputs are main options. Online Document can be written in any script. Online documents may be written in different languages and scripts. Most of the text recognition algorithms are designed to work with a particular script Therefore, an online document analyzer must first identify the script before employing a particular algorithm for text recognition. A manuscript is defined as a graphical form of a writing system. A specific script like Roman may be used by multiple languages such as English, German and French

Keywords: Scripts, language, Analyser.

1. INTRODUCTION

Automatic identification of handwritten script facilitates many important applications such as automatic transcription of multilingual documents and search for documents on the Web containing a particular script. The increase in usage of handheld devices which accept handwritten input has created a growing demand for algorithms that can efficiently analyze and retrieve handwritten data. The web contains text in many languages and scripts or we can say online documents may be written in different languages and scripts. Most of the text recognition algorithms are designed to work with a particular script and treat any input text has being written only in the script under consideration. Fig.1 shows an example of a document page containing six different scripts. . A specific script like Roman may be used by multiple languages such as English, German and French. The six scripts considered in this work, named Arabic, Cyrillic, Devnagari, Han, Hebrew, and Roman, cover the languages used by a majority of the world population (see Fig. 1). Handwriting recognition refers to the ability of a computer to receive intelligible written input. This paper represents

1. Allows the users to write a new script and then save it.
2. Users can write a script and then system can recognise it.
3. Users can view all the scripts saved in the database.
4. Users can delete any/all the saved scripts.

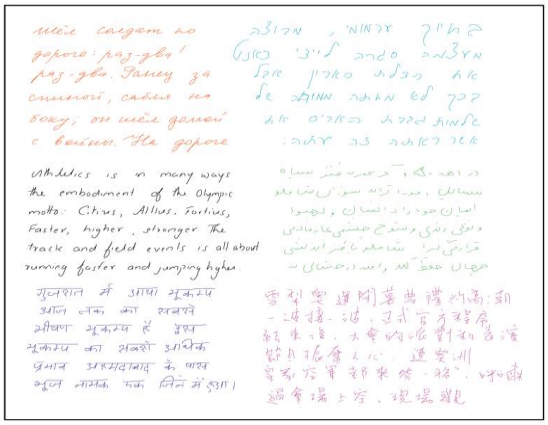

Fig1: multilingual document containing Cyrillic, Hebrew, Roman, Arabic, Devnagari, and Han scripts.

2. Implementation

This paper is implemented in Java Technology. This paper contains basically four modules:

1. Data collection and Preprocessing
2. Feature Detection
3. Recognition

2.1 Data collection and Preprocessing

2.1.1 Introduction: In this module, the data samples are collected and stored in the script folder in order to match up to the input data that has been provided by the user. During the Pre-processing, individual strokes are resampled to reduce the variations in scripts due to different writing speeds.

2.1.2 Description; The control for drawing that is writing the script is provided. User can choose the language and write the script document with several pages and store inside script folder. For one language various scripts can be written and stored. These documents or samples are used for recognition process. When the number of sample increases the error rate for recognition can be reduced. However, the writing styles of people may vary considerably on different writing surfaces and the script classifier may require training on different surfaces. During pre processing, the individual strokes are resampled to make the sampled points equidistant. This helps to reduce the variation in scripts due to different writing speeds and to avoid anomalous cases such as having a large number of samples at the same position when the user holds the pen down at a point. The individual strokes are again resampled to make points equidistant. The resampling distances in both cases were set to 10 pixels. During the lowpass filtering and resampling operations, the critical points in a stroke are retained. A critical point is defined as a point in the stroke where the x or y direction of the stroke reverses (changes sign), in addition to the extreme points Pen-up or pen-down) of the stroke. This module helps us to get the input data from the user and to perform the preliminary analysis of the input data to avoid script variations due to different writing rate.

2.2 Feature Detection: Each sample or pattern that we attempt to classify is either a word or a set of contiguous words in a line. In this module, the general characteristics of the six scripts and the 11 features are taken into account.

The properties of six different scripts used in this project are listed below.

1. Arabic: Arabic is written from right to left within a line and the lines are written from top to bottom. A typical Arabic character contains a relatively long main stroke which is drawn from right to left, along with one to three dots. The character set contains three long vowels. Short markings (diacritics) may be added to the main character to indicate short vowel. Due to these diacritical marks and the dots in the script, the lengths of the strokes vary considerably.
1. Cyrillic: Cyrillic script looks very similar to the cursive Roman script. The most distinctive features of Cyrillic script, compared to Roman script are: 1) individual characters, connected together in a word, form one long stroke, and 2) the absence of delayed strokes. Delayed strokes cause movement of the pen in the direction opposite to the regular writing direction.
2. Devnagari: The most important characteristic of Devnagari script is the horizontal line present at the top of each word, called “Shirorekha”. These lines are usually drawn after the word is written and hence are similar to delayed strokes in Roman script. The words are written from left to right in a line.
3. Han: Characters of Han script are composed of multiple short strokes. The strokes are

usually drawn from top to bottom and left to right within a character. The direction of writing of words in a line is either left to right or top to bottom. The database used in this study contains Han script text of the former type (horizontal text lines).

4. Hebrew: Words in a line of Hebrew script are written from right to left and, hence, the script is temporally similar to Arabic. The most distinguishing factor of Hebrew from Arabic is that the strokes are more uniform in length in the former.
5. Roman: Roman script has the same writing direction as Cyrillic, Devnagari, and Han scripts. We have already noted the distinguishing features of these scripts compared to the Roman script. In addition, the length of the strokes tends to fall between that of Devnagari and Cyrillic scripts.

The features are extracted either from the individual strokes or from a collection of strokes. Here, we describe the features and their method of computation.

1. HorizontalInterstroke Direction (HID):

This is the sum of the horizontal directions between the starting points of consecutive strokes in the pattern. The feature essentially captures the writing direction within a line.

$$\mathbf{HID} = \sum_{i=1}^{n-r} \mathbf{dir}(\mathbf{i}, \mathbf{i}+\mathbf{r}),$$

Where,

dir(i,j)=

$$\begin{cases} +1 & \boldsymbol{Xstart(strock\ i)} < Xstart(strock\ j) \\ -1 & \boldsymbol{otherwise}, \end{cases}$$

Where Xstart(.) denotes the x coordinate of the pen-down position of the stroke, 'n' is the number of strokes in the pattern, and 'r' is set to 3 to reduce errors due to abrupt changes in direction between successive strokes. The value of HID falls in the range [r-n; n-r].

2. Average Stroke Length: Each stroke is resampled during preprocessing so that the sample points are equidistant. The number of sample points in a stroke is used as a measure of its length. The Average Stroke Length is defined as the average length of the individual strokes in the pattern.

$$\text{ASL} = \frac{1}{n}\sum_{i=1}^{n} \text{lenght(strock i)}$$

Where n is the number of strokes in the pattern. The value of ASL is a real number which falls in the range [1.0, R_0] where the value of R_0 depends on the resampling distance used during preprocessing (R_0 = 600 in our experiments).

3. Shirorekha Strength: This feature measures the strength of the horizontal line component in the pattern using the Hough transform. This is the distinguishing feature of the devanagiri Script.

$$\text{ShirorekhaStrength} = \frac{\sum_{\forall\, r, -10<\theta<10} H(r,\theta)}{\sum_{\forall\, r,\theta} H(r,\theta)}$$

Where $H(r,\theta)$ denotes the number of votes in the $(r,\theta)_{th}$ bin in the two-dimensional Hough transform space. The Hough transform can be computed efficiently for dynamic data by considering only the sample points. The numerator is the sum of the bins corresponding to line orientations between -10° and 10° and the denominator is the sum of all the bins in the Hough transforms space. Note that it is difficult to constraint the values of r in the transform space due to variations introduced by sampling and the handwriting itself. The value of Shirorekha Strength is a real number which falls in the range [0.0,0.1].

4. Shirorekha Confidence: Each stroke in a pattern is inspected for three different properties of a Shirorekha. Shirorekha spans the width of a word, always occurs at the top of the word, and is horizontal.

5. Stroke Density: This is the number of strokes per unit length (x-axis) of the pattern. Note that the Han script is written using short strokes, while Roman and Cyrillic are written using longer strokes.

$$\text{Strock density} = \frac{n}{width(pattern)}$$

6. Aspect Ratio: Ratio of the width to the height of a pattern. This feature is more meaningful for word level classification than for the classification of a complete line.

7. Reverse Distance: This is the distance by which the pen moves in the direction opposite to the normal Writing direction. The normal writing direction is different for different scripts as we noted at the beginning of this section.

8. Average Horizontal Stroke Direction: Horizontal Stroke Direction (HD) of a stroke can be understood

as the horizontal direction from the start of the stroke to its end.

9. Average Vertical Stroke Direction: Similar to the Average Horizontal Stroke Direction but in vertical direction.

10. Vertical Interstroke Direction (VID): Sum of the Vertical directions between the starting points of consecutive lines in the pattern.

11. Variance of Stroke Length: Variance in sample lengths of individual strokes within a pattern.

2.2.1 Classifier Design: Experiments were conducted with different classifiers to determine the one which performs the best for the problem in hand. Here, we describe the details of each of the classifiers employed in our experiments.

1. k-Nearest Neighbor (KNN) classifier: The distance between two feature vectors was computed using the Euclidean distance metric. Since the features computed differ drastically in their range of values, a linear transformation was applied to every feature to make their mean zero, and variance unity, in the training set. This transformation is referred to as normalization in the remainder of this paper.
2. Bayes Quadratic classifier: The distribution of the feature vectors for each of the classes was assumed to be Gaussian, and the mean and the covariance matrix were estimated from the training data. A Bayesian classifier, with equal priors was used to classify the test patterns into one of the six script classes. The features were normalized as described before.
3. Bayesian classifier with Mixture of Gaussian Densities: The densities of each of the six classes were assumed to be a mixture of Gaussians. The number of Gaussians within a mixture and their parameters were estimated using the EM algorithm. The classifier itself is similar to the Bayes Quadratic classifier. The features were normalized.
4. Decision Tree-based classifier: The decision tree-based classifier partitions the feature space into a number of sub-regions by splitting the feature space, using one feature at a time (axis-parallel splits). The regions are split until each sub-region contains patterns of only one class with a small number of possible outliers.
5. Neural Network-based classifier: We used a three-layer neural network (one hidden layer) for the script classifier. The input layer contains 11 nodes, corresponding to each of the features in the feature vector. The output layer contains six nodes, corresponding to the six script classes. The number of nodes in the hidden layer was experimentally determined as 25. Increasing the number of hidden nodes above 25 gave little improvement in classification accuracy. During the training phase, the desired output for the node corresponding to the label of the pattern is set to 1, and all other node outputs are set to 0. When a test pattern is fed to a trained neural net, the class corresponding to output node with highest output value is determined to be the result of the classification.
6. Support Vector Machine (SVM): Support vector machines map an n-dimensional feature vector to an m-dimensional feature space (m > n), with the assumption that the patterns belonging to different classes are linearly separable in the m-dimensional feature space. The mapping is implicitly defined by a kernel function. We used the SVM Torch package to conduct the experiments and used the linear, polynomial and radial basis function (RBF) kernels. The classifier using RBF kernel performed best among the three.

2.2.2 Combining Multiple Classifiers: The results of the different classifiers may be combined to obtain better classification accuracy. The results can be combined at different stages in the classification process. We have used a confidence level fusion technique where each classifier generates a confidence score for each of the six scripts. The confidence score is a number in the range ½0; 1_, where 0 indicates that the test pattern is least likely to be of the script associated with the score, while a confidence score of 1 indicates that the test pattern

is most likely to be the corresponding script. The confidence scores generated by the individual classifiers are summed and normalized to the range [0, 1] to generate the final confidence score. The script which has the highest score is selected as the true class. In our experiments, we combined the results obtained from SVM, KNN (k = 5), and neural network classifiers. For SVM classifier, the confidence was generated from the output of the individual (two class) classifiers. The confidence score for the KNN classifier was computed as the proportion of neighbors which belong to the decided class. The output value of the node corresponding to the decided class gives the confidence value for the neural net-based classifier. The combined classifier could attain an accuracy of 87.1 percent on 5-fold cross-validation. The standard deviation of error over the cross-validation runs was 0.3 percent. Table 4 gives the confusion matrix for the combined script classifier which discriminates individual text lines.

2.3 Recognition

2.3.1 Introduction: The feature vector of the input data is matched up with all feature vectors of the different sample scripts. The name of the appropriate script will be displayed if the feature vectors are matched.

2.3.2 Description; In this project the classifier used is Euclidean distance calculation. The distance between two feature vectors was computed using the Euclidean distance metric.

2.3.3 Euclidean Algorithm-Background: One of the oldest and one of the most widely used distance formulas is the Euclidean formula. The formula captures two items and compares each attribute of each item to one another to determine how close, related, they are. In other words, with the formula we can take two users run through each of their characteristics, in our case movie ratings, and determine how similar they are. Once the calculations are done between two users we are presented with a value between 0 and infinity. The larger the value the farther apart both users are to one another (not similar). The Math Equation

$$\sqrt{(p_1-q_1)^2+(p_2-q_2)^2+\cdots+(p_n-q_n)^2}=\sqrt{\sum_{i=1}^{n}(p_i-q_i)^2}.$$

The equation above gives us both the short hand and the expanded version.

3. System Analysis

3.1 Proposed System: The proposed method uses the features of connected components to classify six different scripts (Arabic, Cyrillic, Devnagari, Han, Hebrew, Roman) and reported a classification accuracy of 88 percent on document pages. There are few important aspects of online documents that enable us to process them in a fundamentally different way than offline documents. The most important characteristics of online documents are that they capture the direction of strokes while writing the document. This allows us to analyze the individual strokes and use the additional direction for both script identification as well as text recognition. In the case of online documents, segmentation of foreground from the background is a relatively simple task as the captured data, i.e., the (x; y) coordinates of the locus of the stylus, define the characters and any other point on the page belongs to the background. We use stroke properties as well as the spatial and temporal information of a collection of strokes to identify the script used in the document The advantages of the proposed system are as follows:

1. An online document is the main consideration.
2. The proposed system deals with both spatial and temporal characteristics of the document.
3. Easy segmentation of foreground and background of the online document.
4. Few Error rate.

3.2 Test Case

Many test cases have been designed.

Software testing has three main purposes: verification, validation, and defect and error finding.

Test Case	Description	Action	Desired Result	Actual Result
U1	Checking the functionality for creating new text.	On the left panel select the "script type" & click on new text.	We should be able to create some text in canvas without errors	We are able to draw text after creating on New Text button
U2	Checking the functionality for saving text.	After drawing text , click on Save text button	It should be able to save text in scripts folder	It was able to save text , which was shown as change in no. of script indicated along with script type in left pane
U3	Checking the working of end text button.	After saving the drawn text , click on End text button	The canvas should get cleared	The canvas was all cleared up after prompt
U4	Checking the working of clear text button.	Click on Clear Text button to remove drawn text	The drawn text having flaws should get cleared	The flaw text was cleared from the canvas
U5	Checking the working of menu bar and menu items.	Click on the menus File or Help	We should be able to view how to document or close window	The how to document appeared in notepad an dwe were able to close the project window
U6	Checking the working of Delete All Scripts button	Click on Delete all scripts button to remove saved scripts	All saved scripts should be removed	All scripts were deleted from scripts folder marked by the appearance of no. 0 with the script types in left pane
U7	Checking the efficiency of Recognition	Click on New text Button in Recognition script type ,draw text and click on recognize button	The text should be recognized in one of the saved script types	Text was recognized in following 6 scripts with 87.1 percent efficiency
U8	Checking the working of canvas	Click on canvas and hold down the mouse coursor until 1 stroke is complete	We should be able to draw text on canvas with words and lines	Success ,we were able to draw text on the canvas

U9	Checking the working of Number of points and strokes calculator	While drawing text in canvas look for changes in no. of strokes and points in upper pane	It should present the actual no. of points and strokes	It presented the no of points and strokes in upper pane

3.3 System Design

Design is concerned with identifying software components specifying relationships among components, specifying software structure and providing blue print for the document phase.

3.4 Snapshots

Fig. 1 Selecting Script in VSR

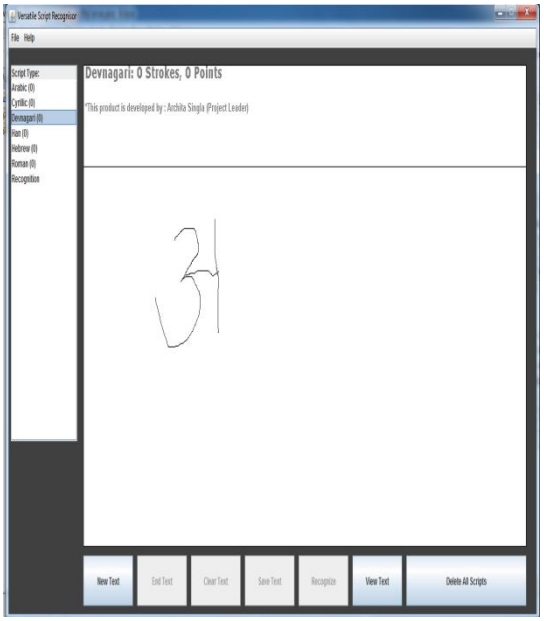

Fig. 2 Creating a new text in Devnagari

This is the second snapshot showing that text for Devnagari script has been written and is saved by clicking on the button "Save Text".

4. Conclusion and Future Work

We have presented a script identification algorithm to recognize six major scripts in an online document. The aim is to facilitate text recognition and to allow script-based retrieval of online handwritten documents. The classification is done at the word level, which allows us to detect individual words of a particular script present within the text of another script. The classification accuracies reported here are much higher than those reported in the case of script identification of offline handwritten documents, although the reader should bear in mind that the complexities of the two problems are different. One of the main areas of improvement in the above algorithm is to develop a method for accurately identifying text lines and words in a document. We are currently working on developing statistical methods for robust segmentation of online documents. The script classification algorithm can also be extended to do page segmentation, when different regions of the handwritten text are in different scripts.

REFERENCES

[1]. Anoop M. Namboodiri, and Anil K. Jain, "Online Handwritten Script Recognition" Ieee Transaction On Pattern Analysis And Machine Intelligence, Vol. 26, no. 1, January 2004.

[2]. A.L. Spitz, "Determination of the Script and Language Content of Document Images," IEEE Trans. Pattern Analysis and Machine Intelligence,

[3]. Anil Kumar Singh and Jagadeesh Gorla, "Identification of Languages and Encodings in a Multilingual Document", in Language Technologies Research Centre

[4]. A.K. Jain and Y. Zhong, "Page Segmentation Using Texture Analysis," *Pattern*

Recognition, vol. 29, pp. 743-770, May 1996.

[5]. U. Pal and B.B. Chaudhuri, "Script Line Separation from Indian Multi-Script Documents," *Proc. Fifth Int'l Conf. Document Analysis and Recognition,* Sept. 1999.

[6]. C.Y. Suen, S. Bergler, N. Nobile, B. Waked, C.P. Nadal, and A. Bloch, "Categorizing Document Images Into Script and Language Classes," *Proc. Int'l Conf. Advances in Pattern Recognition,* pp. 297-306, Nov. 1998.

[7]. J.J. Lee and J.H. Kim, "A Unified Network-Based Approach for Online Recognition of Multi-Lingual Cursive Handwritings," Proc. Fifth Int'l Workshop Frontiers in Handwriting Recognition, pp. 393-397, Sept. 1996.

[8]. C.L. Tan, P.Y. Leong, and S. He, "Language Identification in Multilingual Documents," *Proc. Int'l Symp. Intelligent Multimedia and Distance Education,* Aug. 1999.

[9]. G.S. Peake and T.N. Tan, "Script and Language Identification from Document Images," *Proc. Third Asian Conf. Computer Vision,* pp. 96-104, Jan. 1998.