Multi-lingual Date Field Extraction for Automatic Document Retrieval by Machine

Ranju Mandal\textsuperscript{a}, Partha Pratim Roy\textsuperscript{b}, Umapada Pal\textsuperscript{c}, Michael Blumenstein\textsuperscript{a}

\textsuperscript{a}School of Information and Communication Technology, Griffith University, Queensland, Australia
\textsuperscript{b}Dept. of Computer Science \\& Engineering, Indian Institute of Technology, Roorkee, India
\textsuperscript{c}Computer Vision and Pattern Recognition Unit, Indian Statistical Institute, Kolkata, India

Abstract

Robotic intelligence has recently received significant attention in the research community. Application of such artificial intelligence can be used to perform automatic document retrieval and interpretation by a robot through query. So, it is necessary to extract the key information from the document based on the query to produce the desired feedback. For this purpose, in this paper we propose a system for automatic date field extraction from multi-lingual (English, Devnagari and Bangla scripts) handwritten documents. The date is a key piece of information, which can be used in various robotic applications such as date-wise document indexing/retrieval. In order to design the system, first the script of the document is identified, and based on the identified script, word components of each text line are classified into month and non-month classes using word-level feature extraction and classification. Next, non-month words are segmented into individual components and labelled into one of text, digit, punctuation or contraction categories. Subsequently, the date patterns are searched using the labelled components. Both numeric and semi-numeric regular expressions have been used for date part extraction. Dynamic Time Warping (DTW) and profile feature-based approaches are used for classification of month/non-month words. Other date components such as numerals and punctuation marks are recognised using a gradient-based feature and Support Vector Machine (SVM) classifier. The experiments are performed on English, Devnagari and Bangla document datasets and the encouraging results obtained from the system indicate the effectiveness of the proposed system.

Keywords: Robot vision, Robot reading, Robot retrieval of document, Date-based indexing, Handwritten date extraction, Date spotting, Multi-lingual documents

1. Introduction

Robotic intelligence has been applied in many fields to perform tasks which were previously performed by humans. At present various robot systems have been designed in many areas such as Engineering, Medical, Industrial applications, etc. Although there is a big effort in the robotics community in these areas [35, 1], still to date, robots have been made little use of human readable text. To our knowledge, only a few pieces of work have undertaken towards
realising a librarian robot to search and retrieve a book requested by a user [16, 29, 31]. The operation of such a robot starts when the user requests a book by its name or code, either through the Internet or by voice. The robot then locates the book in the library, extracts it and then takes it to the user. Some works have also been proposed towards a robotic system that is capable of detecting and reading wild text, a rich source of semantic information indigenous to man-made environments [15].

Nowadays huge collections of documents are available and automatic indexing or retrieval of relevant documents from these huge collections needs lot of human effort. For example, date-wise sorting of administrative documents of an organization may require a lot of manual effort and also it is time consuming. So it is necessary to think of robotic applications to perform such tasks. Keeping this in mind, in this work we propose a date-based document indexing system for a robotic application. To our knowledge, there is no quality work on the application of robotics to document analysis such as document/form reading, document indexing, etc. by a robot. Thus, the objective of this paper in this special issue is to attract the attention of the robotics researchers in this area.

In this paper we propose a system for automatic date field extraction from multi-lingual (English, Devnagari and Bangla scripts) handwritten documents. Date is a useful piece of information and it could be used as a key in various applications, for e.g., date-based document searching and indexing of document repositories such as administrative documents, historical archives, postal mail etc. Also, automatic extraction of date information is a challenging task due to different date patterns (Numeric and Semi-numeric dates consisting of different lengths), writing styles of different individuals, touching characters and confusion of classification during identification of numerals, punctuation and texts, etc. In multi-lingual and multi-script countries such as India, retrieval of multi-script documents using the date pattern can be very effective. An Indian state generally uses three official languages. For example, the West Bengal State of India uses Bangla, Devnagari and English as official languages. Hence, a single document may contain one or more of these three scripts. Fig. 1 shows an Indian handwritten Bangla postal document containing an English (Roman) date. English script is widely used in India with popular handwritten Indic scripts such as Bangla and Devnagari in a single document. Often, date information is written using English numerals in these scripts. Moreover, Bangla documents having Devnagari and English (Roman) script is also common. Because of multi-lingual behaviour, our date field extraction method consists of three major tasks namely: script identification, month word detectors, and numerical field and date pattern extraction. The proposed tri-lingual date extraction method can handle five cases such as Devnagari documents with only Devnagari or English date fields, Bangla documents with only Bangla or English date fields and English documents with English date fields. Two types of date field patterns (Numeric and Alpha-numeric) are considered for all the above cases. Hence, the proposed date extraction process from such documents will be very useful in searching and interpreting documents.

Because of the different writing formats of dates, script analysis is a necessary and important task in date field extraction from multi-script documents. We have shown some examples of handwritten documents containing date information in Fig. 2. It is important to note that, the date patterns appear in different formats in a document. Some of these formats of a single English
Date are 12/03/2012 or 12th March, 2012 or March 12, 2012 or 12-03-2012 or 12.03.2012 or 12.03.12, etc. Examples of Bangla dates are ১৫/০৭/১৯৯০ in dd/mm/yyyy format, ১লা বৈশাখ ১৪১১, ২রা আষাঢ় ১৪১২, ৪ই আষাঢ় ১৪১৪, ২১ শে বৈশাখ ১৪১৪, ৩রা জানুয়ারী ১৯৯০, ২রা জানুয়ারী ১৯৯০ in dd month yyyy format and ১৬-০৭-১০ in dd-mm-yy format, etc. Some sample formats of Devnagari dates are १२/०८/२००९ in dd/mm/yyyy format, etc.

Figure 2: Examples of handwritten and printed documents containing numeric and semi-numeric date fields in Bangla (a) and English (b,c,d). Numeric and semi-numeric date fields are marked with blue and red rectangles, respectively.

1.1. Related Works

Date field extraction from multi-lingual documents consists of three major tasks: script identification, month word identification, numerals and punctuation (e.g., slash ‘/’, hyphen ‘-’ and period ‘.’) identification. To the best of our knowledge, there is no work on date field extraction from multi-script documents. However, to get the idea about the state-of-the-art work in script identification, word spotting, numerical field extraction and date field recogni-
1.1.1. Script Identification

The literature is replete with research on script identification for printed and handwritten (offline/online) documents. The script identification work previously proposed can be categorized into three types such as block/paragraph level, line level and word level. Spitz [27] proposed a two-stage approach for automatic identification of script as well as language from printed document images. First, the script is classified into two initial classes (Han-based or Latin-based) using vertical position distribution of upward concavity information. Distribution of optical density information is used to determine the language of the Han script class (Chinese, Japanese, and Korean). Latin-based languages are determined using information based on character shape codes. A connected component-based script and language identification method for handwritten documents has been proposed by Hochberg et al. [6]. A set of heuristic features are extracted from each component and linear discriminant analysis is used for classification of each possible pair of scripts in the dataset (Arabic vs. Chinese, Arabic vs. Cyrillic, etc.).

Roy and Majumdar [21] proposed a method for script separation of postal documents using features based on geometric pattern, busy-zone and topology. A Neural Network was used in this work for classification. Three scripts (Bangla, Devnagari and English) were considered in their work. In [19] Roy et al. proposed a technique towards identification of handwritten Roman and Persian scripts. A set of 12 features based on fractal dimensions, position of small components, topology etc. were used. Four classifiers (SVM, Nearest Neighbour, MLP and MQDF) were applied for the classification task and a comparative analysis was presented in the paper. In another work Roy et al. [20] proposed an algorithm for script identification at the document level from Indian handwritten documents written in six popular scripts: Bangla, Devnagari, Malayalam, Urdu, Oriya and Roman. Fractal-based features, a component-based feature and Topological features were used to extract a total set of 46 features in their work. A Multi-layer Perceptron (MLP) Neural Network was used for identification.

Zhou et al. [36] proposed a method for identification of Bangla/English scripts. The method was based on the connected component profile extracted from a block of handwritten addresses. Singhal et al. [26] proposed a technique for classification of four different scripts viz. English, Devnagari, Bangla and Telugu. Rotation invariant textures feature using multi-channel Gabor filter were extracted and multi-prototype classifiers were used for their classification task. Sarkar et al. [25] proposed a script separation technique for Bangla and Devnagari script mixed with Roman script. A set of holistic features were extracted at the word level and Multi-Layer Perceptron (MLP)-based classifier was used in this work.

1.1.2. Word Spotting

Significant work has been undertaken [17, 4, 5] in the area of word spotting to make handwritten text available for searching and browsing. Word spotting is becoming popular because of the low computational cost in comparison to
the transcription of entire text. Rath and Manmatha [17] proposed a classical method for word spotting using profile-based features and Dynamic Time Warping (DTW). A Recurrent Neural Network-based approach was proposed in [5] to make handwritten documents available for word-based searching and indexing. Neural Networks and CTC Token Passing algorithms were used for the word spotting task.

Hidden Markov Model (HMM)-based methods are extensively used for modelling handwritten text, word spotting, etc. In [4], Fischer et al. proposed a learning-based word spotting system that uses HMM sub-word models to spot keywords. The proposed lexicon-free approach can spot arbitrary keywords from the handwritten text. In [18], a HMM-based method was employed for word spotting from handwritten documents. Local Gradient Histogram (LGH) features were used in this work.

1.1.3. Numerical Field Extraction

Few research works have been published for automatic form field extraction from handwritten documents [2, 30, 8, 3]. Recently, field-based information retrieval has received more popularity than the recognition of full handwritten documents. A segmentation-driven recognition-based approach was proposed by Chatelain et al. [2] for automatic extraction of numerical fields from handwritten incoming mail documents. Different feature sets such as contextual/morphological, chain code and statistical/structural were extracted. A multi-layer perceptron is trained over each feature set and measurement level combination of classifiers is achieved. The main objective of this work is to localize the fields of interest by rejecting the major part of the document. The false alarm rate has been slightly decreased by the use of combination of classifiers.

Thomas et al. [30] proposed a HMM-based classification model for alphanumeric sequence recognition. The handwriting modelling was text line oriented and a HMM-based technique was used as a probabilistic tool in handwriting sequence modelling [33]. Here, the global line model is based on HMMs, which is employed for dual representation of relevant and irrelevant information. The shallow parsing of text lines allows the fast extraction of information. To extract the desired numerical sequence, a syntactical analysis was performed on each line of text. In [3] Chatelain et al. proposed a method for automatic extraction of numerical fields (ZIP codes, phone numbers, etc.) from incoming mail documents aiming at locating isolated and touching digits among textual information. A syntactical analysis is then performed on each line of text in order to filter the sequences that represent a particular syntax known by the system.

Koch et al. [8] proposed a method for automatic extraction of numerical fields from handwritten documents. The approach exploits the known syntactic structure of the numerical field to be extracted, combined with a set of contextual morphological features to find the best label of each connected component. A HMM-based syntactic analyser on the overall document is employed to localize/extract fields of interest. The localisation of numerical fields is performed using the Viterbi algorithm. Most of the papers mentioned here deal with alphanumeric string extraction. Therefore, spotting-based approaches outperformed OCR-based systems for such information extraction systems in handwritten documents.
1.1.4. Date Field Recognition

Handwritten date information processing from scanned documents still remains very challenging and it is hard to propose an algorithmic method for automatic segmentation and recognition. There are very few published research works available on automatic segmentation and recognition of date-fields and this existing work was developed for specific documents such as bank cheques. These methods cannot handle other documents of different date formats. Suen et al. [28] proposed an approach focusing on one of the most challenging parts of a cheque recognition system, i.e., the segmentation and recognition of the date written on the bank cheques. First, the method segments a date image by using the separator information. Two separators are then used to detect the ‘Year’, ‘Day’ and ‘Month’ zones based on shape and spatial features. Next, numeric and non-numeric month fields are recognised by a connected digit recogniser and a cursive word recognizer. The recognition results are finally sent to a parser, which is used to interpret acceptable results and to reject invalid ones.

Xu et al. [33] described a knowledge-based segmentation system for handwritten dates on bank cheques. The knowledge derived from writing style information, also syntactic and semantic constraints are utilized, and different knowledge sources are adopted at different stages. In another work [34], Xu et al. describes a system for automatic segmentation and recognition of handwritten dates on Canadian bank cheques. A segmentation-based strategy is adopted in this system. In order to achieve high performance in terms of efficiency and reliability, a knowledge-based module is proposed for date segmentation and a cursive month word recognition module is implemented based on a combination of classifiers.

Our present proposed work moves a step further in document interpretation and uses the recognition labels of alpha-numeric characters of three languages (Bangla, Devnagari and English) to locate the date fields in Indian multi-lingual documents. Recently, we proposed a method for date field extraction from handwritten English documents [9]. The present work is an extended version which deals with date extraction from multi-script (English, Devnagari and Bangla) handwritten Indian documents. To the best of our knowledge, there is no work available on date extraction in printed/handwritten multi-lingual documents. Date pattern detection and interpretation in handwritten documents is challenging due to the unconstrained nature of handwriting of different individuals, touching numerals, different patterns of a single date etc.

1.2. System Overview

A multi-stage approach has been proposed here for date field extraction. In the first stage, scripts are identified using foreground and background information. A water reservoir-based technique is employed here to extract foreground and background information. Top and bottom reservoir information of words is used to segment the words into primitive segments and those segments are classified using SVMs as Bangla, Devnagari or English primitive segments. Based on the majority of classified primitive segments, a script is identified. Once the script is identified, the system is trained with the respective models. For each script, the system is trained with two (Month and digit) models for classification in two stages. In the second stage, month and non-month handwritten word blocks are separated. For this purpose, word blocks are extracted using morphological operations and the segmented word blocks are classified into month and
non-month classes using word block-level feature analysis. The third stage per-
forms component analysis for each non-month handwritten word block. Isolated
digits, punctuations and alphabets are identified using component-level feature
analysis and recognition. The components with low recognition confidence are
analysed further for touching segmentation [23]. We have used a DTW-based
technique for word block classification and 400-dimensional gradient-based fea-
tures and SVMs are used in component level classification. In the final stage,
numeric and semi-numeric (contains month field as text string) date patterns
are explored from the sequence of the labelled components. To do so, candi-
date lines are selected first using a voting approach and next regular expression
analysis is used to detect the date patterns.

The organization of the rest of this paper is as follows: In Section 2, we
discuss our script identification method. The proposed methodology of date
field extraction is detailed in Section 3. The experimental results have been
described and analysed in Section 4. Finally, conclusions are drawn in Section
5.

2. Script Identification

A global histogram-based Otsu binarization [11] method has been applied to
convert the grey-scale document images into binary images. Next, a smoothing
algorithm [14] has been used to remove noisy pixels and irregularities. Our date
retrieval approach searches the date patterns in text line images. Hence, the bi-
nary document is segmented into individual text lines using a line segmentation
algorithm [22] and segmented lines are used for experimentation.

For line segmentation, at first the water reservoir-based [12] information is
used to determine the height of text lines present in a document. The water
reservoir is a metaphor to illustrate the cavity region of a component. The water
reservoir principle is as follows. If water is poured from a side of a component,
the cavity regions of the background portion of the component where water will
be stored are considered as reservoirs of the component. We have then applied
the horizontal Run Length Smearing Algorithm (RLSA) on the input image.
The threshold of RLSA is computed based on the height of text lines. Next,
the foreground components of the RLSA applied image are eroded to generate
a few seed components from the individual words of the document. Actually,
the central gravity of individual words of a text line is represented by the seed
components. The erosion method also helps to reduce the touching effect of
character modifiers and makes the line segmentation task easier. To find the
upper and lower boundary information of a text line, the erosion technique is
also applied on the background region of the image. Finally, the positional
information of the seed components and the boundary information are used to
segment the individual lines.

Both foreground and background information is used for script identifica-
tion. A water reservoir-based [12] technique is employed here for identification
of English, Devnagari and Bangla scripts. Top and bottom reservoirs are used
for the segmentation of the words into primitives. By top (bottom) reservoirs
of a component we mean the reservoirs obtained when water is poured from the
top (bottom) of the component. A bottom reservoir of a component is visual-
ized as a top reservoir when water will be poured from the top after rotating
the component by 180°. Reservoir-based primitive segments (foreground) and
reservoir blobs (background) are used as primitive components of these three scripts (see Fig. 3). A word is segmented into a few primitive components. Segmentation points for primitives of a word are nothing but the bottommost (topmost) points of top (bottom) reservoir in a word. For primitive segmentation, the image is segmented vertically at the segmentation points. The reservoir blobs (filled with a red colour in Fig. 3) are also extracted from top and bottom reservoirs. Primitives segments and the reservoir blobs are used for script identification.

Now, all the primitive segments and reservoir blobs in a script are classified separately for script identification. A 400-dimensional gradient-based feature [13] has been extracted from the primitive segments and the reservoir blobs, which have subsequently been fed to an SVM [32] (Gaussian kernel with Radial Basis Function) for script identification. The primitive segment classifier is trained with the training data consisting of English, Devnagari and Bangla primitive segments. Likewise, a reservoir blobs classifier is also trained. Next, recognised primitive segments are counted as English, Devnagari or Bangla primitive segments and the recognised reservoir blobs are also categorised into three script classes. The resultant script is identified on the basis of a majority voting of these elements. Initially we used primitive segments and reservoir information separately on each word for its identification. Later such individual identification results are combined jointly to get final results. For an example of combinations, let an input Devnagari word sample contain 6 primitive segments and 4 reservoirs. Out of these 6 primitive segments of the input word, let 4 primitives be classified as Devnagari and 2 primitives as Bangla. Also, out of the 4 reservoirs of the input word, let 2 reservoirs be classified as Devnagari, 1 reservoir be classified as Bangla and 1 reservoir be classified as English. Since, 4 primitive segments and 2 reservoirs are classified as Devnagari, hence we consider 6(4+2) is the combined result of primitive segmentation and reservoir information for Devnagari. Similarly, we consider 3(2+1) is the combined result of primitive segmentation and reservoir information for Bangla as 2 primitive segments are classified as Bangla and 1 reservoir is classified as Bangla. Finally, the word is recognised as Devnagari as the majority of elements (6) of the input word are classified as Devnagari. The feature extraction technique and classifier used in our method is described below.

**Feature Extraction and Classification:** We compute 400-dimensional gradient-based features at the word component level using the following method. These features help us to get higher segmentation confidence using SVMs.

**400-dimensional gradient feature:** The grey-scale local-orientation histogram of the component is used for 400-dimensional feature [13] extraction. To obtain 400-dimensional features we apply the following steps.

- At first, size normalization of the input binary image is done. Here we normalize the image into 126 × 126 pixels.
- The input binary image is then converted into a grey-scale image by applying $2 \times 2$ mean filtering 5 times.
- The grey-scale image is normalized next so that the mean grey-scale becomes zero with maximum value 1.
Next, the normalized image is segmented into $9 \times 9$ blocks.

A robust filter is then applied on the image to obtain the gradient image. The arc tangent of the gradient (strength of gradient) is quantized into 16 directions (an interval of $22.5^\circ$) and the strength of the gradient is accumulated with each of the quantized direction. By strength of Gradient
\( (f(x,y)) \) we mean \( f(x, y) = \sqrt{(\Delta u)^2 + (\Delta v)^2} \) and by direction of gradient \( (\theta(x,y)) \) we mean \( (\theta(x,y)) = \tan^{-1}(\Delta u/\Delta v) \) here \( \Delta u = g(x + 1, y + 1) - g(x, y) \), \( \Delta v = g(x + 1, y) - g(x, y + 1) \) and \( g(x, y) \) is a grey-scale value at an \((x,y)\) point.

- Histograms of the values of 16 quantized directions are computed in each of \(9 \times 9\) blocks.
- Finally, \(9 \times 9\) blocks are down sampled into \(5 \times 5\) by a Gaussian filter. Thus, we get \(5 \times 5 \times 16 = 400\) dimensional feature.

**Classifier:** SVM is a popular classification technique which can successfully be applied to a wide range of applications [32]. So, in our experiments, we have used an SVM as the classifier. SVMs are defined for two-class problems and they look for the optimal hyper plane which maximizes the distance, the margin, between the nearest examples of both classes, named support vectors (SVs). Given a training database of \(M\) data: \(x_m|m = 1, ..., M\), the linear SVM classifier is then defined as: \(f(x) = \sum_j \alpha_j x_j + b\) where \(x_j\) are the set of support vectors and the parameters \(\alpha_j\) and \(b\) have been determined by solving a quadratic problem. The linear SVM can be extended to various non-linear variants, details can be found in [32]. In our experiments, the Gaussian kernel SVM outperformed other non-linear SVM kernels, hence we are reporting our recognition results based on the Gaussian kernel only. The hyper parameters of SVM are set as follows; kernel type = RBF, \(\gamma = 1.5\) and \(C = 1\). The best results have been achieved by setting the above values of these parameters which are tuned using a validation process. The Gaussian kernel is of the form:

\[
k(x, y) = e^{-\gamma\|x - y\|^2}
\]

### 3. Extraction of Date Field

A spotting-based approach has been proposed here to find the date field in a document. Our date retrieval approach searches the date patterns in individual text lines of a document. First, different components of a date such as textual month, numeral, punctuation etc. are found and this information is used to extract the date patterns. The different types of date formats are searched and the approach performed to locate those components, as well as the complete date field, are described in this section.

#### 3.1. Date Components

A date field in documents may have four types (Numeral, textual month, punctuation and contraction) of components. Date patterns can be classified into two categories (numeric and semi-numeric dates) on the basis of different component types.

**Numeric date:** A date field consisting of numerals and punctuations (Examples of numeric dates are 12/05/2010, 2/5/2010, 2-5-10 in English; ১২/০১/১৪১৮, ১-৭-১১ in Bangla; जैन/१२/२००९, ३१-१२-११ in Devnagari etc.) are considered here as numeric date fields. It has been observed from the dataset that the total number of components in a numeric date field can vary from 6 to 10 (If a date is written as 1/1/14 then the number of components of this date is 6. If a date is written
as 01/01/2014 then the number of components will be 10. The following date regular expression represents the valid formats of a numeric date:

\[(d|dd)/(.-)(d|dd)/(.-)(dd|dddd)\]

where, d represents a digit. A date has three parts or fields: date field, month field and year field. A complete numeric date field consists of a single digit or double digit date information, single digit or double digit month information and double digit or four digit year information. Moreover, a numeric date field must have 2 punctuation marks to separate day, month and year information.

**Semi-numeric date:** Other date fields that consist of textual month (Examples are January, জানুয়ারী and जनवरी in English, Bangla and Devnagari scripts, respectively), digit and contraction (Examples are st, nd, rd in English; আঁ, দ্বারা, তৃতীয় in Bangla; अ, द्वारा, तीसरा in Devnagari etc.) are considered as semi-numeric dates. Examples of semi-numeric dates are ২১ এপ্রিল, ২০১০; ২০১০ এপ্রিল ২১ in Bangla; २१ एप्रिल, २०१२ in Devnagari and 31st March, 2011 in English etc. For semi-numeric date field extraction, we search for the following regular expressions:

\[(md|mdd)(.,-)(dd|dddd)\] and \[(d|dd)(contraction)(month)(.,-)(dd|dddd)\]

where, m represents a month field and d represents digit or numeral. There are two types of sequence for semi-numeric date fields depending on the position of the textual month field (Examples of date with month information in the middle position: 15th December, 2012 and month information in the starting position: December 15, 2012). In a semi-numeric date pattern, textual month information may be in the front or in the middle of the sequence.

### 3.2. Month/Non-month Identification

A conventional profile-based feature extraction and classification process has been used to identify a word as a textual month. Here, the system performs classification on the extracted words from a line. The proposed classification technique works as a two-class problem: month and non-month word block identification. To train the classifier, data sets with different types of month formats (Examples are: জানুয়ারী, জানুয়ারী, জানুয়ারী in Bangla; जनवरी, जनवरी in Devnagari and JANUARY, January, JAN in English etc.) that appear in date patterns of Bangla, Devnagari and English documents are used. Three different training datasets have been used here to train the classifier for the three different scripts. Once the script is identified, the respective training model is used for this purpose. The features and matching steps used for this month identification process are explained as follows.

**Profile features:** Word profile-based information is used here to determine word-level features. Four components such as vertical projection profile, upper profile, lower profile and vertical crossing (vertical ink transition) have been extracted as discussed in [17] for word-level features. The vertical projection profile is the summation of pixels vertically in an image. The distance between the upper boundary and the closest foreground pixel is considered for the upper profile and the distance between the lower boundary and the closest foreground
pixel is considered here for the lower profile features. Vertical crossings are the number of foreground and background transitions in an image. The value of all the profile features stated above is normalized to the range [0-1]. Profile features for a sample word are presented in Fig. 4.

![Figure 4: Profile features](image)

**Classification of Month Words:** The DTW approach presented here is similar to the method described in [17]. A DTW-based model is used for identifying month/non-month words in this work. The similarity between two month sequences is measured using the DTW technique. The sequences are “warped” non-linearly in the time dimension to determine a measure of their similarity independent of certain non-linear variations in the time dimension. This technique has been widely used in many applications such as speech, signatures, robotics etc. A word is represented as 4 sequences, which are computed from different profiles of a word as described above. The DTW-based technique for measuring similarity between two sequences uses the Sakoe-Chiba band [24] to speed up the computation. The width of the word is used as a pruning criterion, i.e. the width of the word image must not be more than double the width of the other.

Here DTW is used on two signals of 4 sequences of features \( f_k \), where \( 1 \leq k \leq 4 \): the upper and lower profile features, vertical projection profile and background to foreground transition of words. The DTW distance between two signals \( I_1 \) and \( I_2 \) is calculated using a matrix \( D \). Where

\[
D(i, j) = \min \left( \begin{array}{ccc} D(i, j - 1) \\ D(i - 1, j) \\ D(i - 1, j - 1) \end{array} \right) + d(x_i, y_i) \tag{1}
\]

\[
d(x_i, y_i) = \sum_{k=1}^{4} \left( f_k(I_1, i) - f_k(I_2, j) \right)^2 \tag{2}
\]

The matching distances from 4 features of a word are added to get their cumulative distance and this cumulative distance is considered as the final match-
ing cost of the word. Finally, this matching cost is normalized by the length of the warping path. DTW distance and a Nearest Neighbour-based classification technique have been used here for month field spotting. Fig. 5 shows text lines that are classified as month and non-month blocks.

```
(a) Should reach before 11th May, 2012 (Sunday)

(b) On 13th May, 2012 (Sunday).

(c) ২১ মার্চ ২০১২ তারিখে প্রণয়ন হয়েছে।

(d) ২২ আগস্ট ২০১২ তারিখে প্রণয়ন হয়েছে।

(e) the blend to become effective August 16, 1971.
```

Figure 5: Text lines showing detection of month blocks (a,b) English handwritten (c) Bangla handwritten (d) Devnagari handwritten (e) English printed. Here, month, non-month blocks are marked with pink and green, respectively.

### 3.3. Character Level Component Identification

The words which are classified as non-month in the earlier stage are considered here. Connected component analysis is employed to segment the non-month words into different components, and component-wise classification is undertaken to extract the character/digit/punctuation components from these non-month words. For this purpose, connected components are fed to a component level classification stage. A 400-dimensional gradient-based feature [13] has been used for processing the character components, which have subsequently been fed to an SVM [32] (Gaussian kernel with Radial Basis Function) for classification of character components in a word. The details of the feature extraction and classification processes are presented in Section 2. Through the classification process, the components are mainly classified into “digit”, “punctuation”, “contraction” or “text” label (See Fig. 6). There are some components which might be touching and these components could not be classified properly at this stage. Hence, the components with high recognition confidence are accepted and directly considered for date pattern matching. The rest of the components with low confidence are selected for touching component segmentation analysis. Some isolated characters which are not identified properly will be taken care of in the touching character handling step explained in the next subsection.

### 3.4. Touching Character Segmentation

There may exist touching digits/characters in a component. Components with a low confidence score at the earlier stage of recognition are considered as
touching and chosen for segmentation. Here, we use a dynamic programming-based touching character segmentation scheme [23]. First, we find different cavity regions from the touching characters. The cavity regions are obtained using the Water Reservoir concept [12]. We use Top-Bottom reservoir analysis to find the cavity regions in a touching component. A set of candidate segmentation points is obtained from these regions using cavity region analysis. Next, the touching component is segmented into these candidate points to find different sub-images. Using dynamic programming, the recognition confidence of sub-images is analysed and the optimum segmentation path is found. Finally, based on the segmentation lines, the touching component is segmented. This approach segments touching digits in most of the cases. In Fig. 7, the circle shows the segmentation result of two touching digits 2 and 0.

3.5. Date Pattern Search

Text lines, with their four different types of recognized components (month, digit, punctuation, text), are considered here for date pattern detection. We have divided this approach into two parts: candidate line selection and date
pattern search.

3.5.1. Candidate Line Selection

The text lines that contain labelled months, digits and punctuations are selected here. For this purpose, we compute the total number of digits, punctuation marks, and month strings of a text line. Depending on the value of the individual counts, we proceed to search the date patterns in that text line. Valid date patterns in our experimental data set should contain at least 6 (examples are ৪-৬-৯০ in Bangla; ३-४-१२ in Devnagari ; ‘Jan 5th, 12’ etc.) elements. If the total number of date elements in a line is greater or equal to 6 then we consider this line as a candidate line and it is used for date pattern searching.

3.5.2. Date Pattern Searching

The components in each candidate text line are sorted in the left to right direction using the component’s position (Centre of Gravity) and the positions of punctuation, digit and month text are noted. The date patterns are searched next using the sequence of labelled components. In our approach, we consider two different date patterns for searching, namely: numeric and semi-numeric patterns.

Numeric date searching: For numeric date extraction (e.g. ১২-০৩-২০১২, ৪/৭/১০, 15/08/2012, etc.), a sub-sequence of components is matched with the numeric date regular expression (d represents digit):

\[(d|dd)(/.-)(d|dd)(/.-)(dd|dddd)\]

In our date searching algorithm we first find the position of the two punctuation marks. If we get one or two consecutive digits to the left of a left punctuation mark, one or two digits in the middle of these two punctuation marks and two or four digits on the right of a right punctuation mark, we consider this sequence as a valid numeric date field. Devnagari numerals (०-९) and Bangla numerals (০-৯) are used along with English numerals for Devnagari and Bangla script, respectively. This is because people may use English numerals to write Bangla and Devnagari dates.

Semi-numeric date searching: For semi-numeric date fields (e.g. ২১ েশ জুলাই, ২০১০, २३ शे जुलाई, २०१२, 31st March, 2011) extraction, the following semi-numeric regular expressions are searched:

\[(md|mdd)(.,-)(dd|dddd)\] and \[(d|dd)(contraction)(month)(.,-)(dd|dddd)\]

We find the entire pattern in the sequence of line components for matching with any of the above date patterns. In a semi-numeric date pattern, textual month information may be in the front or in the middle of the sequence. 44 different types of textual months such as months in upper case (JANUARY), lower case (January) and short form (JAN, Jan) are considered here for English script. 24 types of month [12 English months written in Devnagari script (e.g. जनवरी, फाल्गुन, मार्च) and 12 Devnagari months (e.g. वैशाख, ज्येष्ठ, आषाढ़)] are considered as
Devnagari month components. 24 types of Bangla month [12 English months written in Bangla script (e.g. জানুয়ারী, ফেব্রুয়ারী) and 12 Bangla months (e.g. টাকা, পৌষ)] are considered here. Four types of contractions (st, nd, rd, th) are used after day information in English dates. In Bangla script, 5 contractions (রা, ঠা, ই, লা, েশ) and in Devnagari script four contractions (ला, रा, शे, इ) are used.

4. Results and Discussion

4.1. Data set

To the best of our knowledge, there exists no standard date dataset consisting of handwritten date samples to evaluate date extraction methods. Hence, different datasets are used for the training of classifiers at different levels. Table 1 shows the details of training and test data used in our experiments. Please note that training and test datasets are different in our experiments.

Training dataset: Row 3 of Table 1 shows the number of Bangla, Devnagari and English handwritten words, which are used to train the SVM classifier at the script identification level. This dataset was collected earlier in our lab for handwriting recognition. Primitive segments and reservoir blobs are extracted from these handwritten training words and are used to train the classifier for identification of scripts. Row 4 of Table 1 shows the month training data. This month data contains different formats (e.g. JANUARY, January, JAN and Jan etc. as mentioned in Section 3.5.2) and collected from 80 individuals. Please note those 44 classes of English months written in alphabetic letters (e.g. JAN-UARY, January, JAN and Jan etc.) and 24 classes each for Devnagari and Bangla scripts are considered in our experiments and used to train the classifier for month identification. Numerals, punctuation etc. are used to train the classifier for component level identification to detect the date. Rows 5-7 of Table 1 show the actual volume of data used to train the classifier at the component level. English digits from the MNIST [7] dataset of handwritten digits have been used for training the classifier at the component level for all scripts. Please note that, as we mentioned earlier, English digits are sometimes used in Bangla and Devnagari dates, hence we used the MNIST data for the training of all the three scripts along with their individual script digits. For example, in Bangla and Devnagari script, we also used Bangla and Devnagari numerals for training along with English MNIST numerals. Bangla and Devnagari numeral data was collected earlier in our lab for postal automation work and used here for training. Here, at least 100 samples per numeral class have been considered for English, Devnagari and Bangla numeral training datasets.

Test dataset: Handwritten text lines of a single script, as well as multi-script data were collected from 60 individuals of different professions and are considered here for testing. Row 9 of Table 1 shows the total number of handwritten lines used from these three scripts for testing purposes.

4.2. Experimental Results

Detailed experimental results of different modules of our system are provided below. Please note that all the experiments are performed using a PC
Table 1: Experimental data set

<table>
<thead>
<tr>
<th>Types of Data</th>
<th>Bangla</th>
<th>Devanagari</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word</td>
<td>2035</td>
<td>2079</td>
<td>1935</td>
</tr>
<tr>
<td>Month</td>
<td>2520</td>
<td>1900</td>
<td>2570</td>
</tr>
<tr>
<td>Numeral</td>
<td>3651</td>
<td>2137</td>
<td>MNIST dataset[7]</td>
</tr>
<tr>
<td>Punctuation</td>
<td>904</td>
<td>904</td>
<td>904</td>
</tr>
<tr>
<td>Contraction</td>
<td>604</td>
<td>578</td>
<td>730</td>
</tr>
</tbody>
</table>

Test Data

| Handwritten Line | 1100 | 820 | 1240 |

with Intel core 2.5 GHz processor and 4 GB of RAM.

Results on script identification: Our script identification experiments were performed to select the features. The computed script identification results using two features on the experimental dataset are shown in Table 2. It shows the SVM-based ($\gamma = 1.5$) 5-fold cross-validation test accuracy for gradient and Gabor filter-based features. The Gabor filter-based feature computed here is the same as the one explained in [10]. Table 2 shows how primitive segments (foreground) and reservoir blobs (background) information individually contribute to the script identification task. It also shows the performance of both the features in this experiment. Gradient features are used finally for script identification as we noted that gradient features outperformed Gabor in SVM-based 5-fold cross validation experiments. Accuracy rates of 73.80% and 65.33% have been obtained for gradient and Gabor, respectively when primitive segment information is used. Script identification using only reservoir blob information produced a 60.84% and 53.89% accuracy for gradient and Gabor features, respectively.

Table 2: SVM-based 5-fold cross validation results for script identification (accuracy based on Primitive Segments and Reservoir Blobs)

<table>
<thead>
<tr>
<th>Feature based on</th>
<th>Primitive Segments(%)</th>
<th>Reservoir Blobs(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gradient</td>
<td>73.80</td>
<td>60.84</td>
</tr>
<tr>
<td>Gabor</td>
<td>65.33</td>
<td>53.89</td>
</tr>
</tbody>
</table>

The combined majority of primitive segments and reservoir blobs information is used for script identification in this proposed work. The results obtained from word-wise script identification for English, Devanagari and Bangla scripts are presented in Table 3. The accuracy obtained for English and Bangla scripts is encouraging. However, 7.15% and 3.12% of Devanagari script is confused with Bangla and English scripts, respectively. Similarity of alphabets as well as the basic script structure between Bangla and Devanagari scripts lead to a 7.15% confusion. If there is a tie in the total number of the recognition results during component classification, the script cannot be determined, and hence it is rejected. But line-wise script identification gives more accuracy than word-wise script identification when the same identification technique is applied. The results of line-wise script identification for these three scripts is presented in Table 3.
The script identification technique proposed here outperformed previous work [21]. Table 5 shows a comparative study with the previously proposed script identification technique. The test dataset used here for the comparison of performance is the same as the one used in [21].

### Table 3: Word-wise script identification results (accuracy is given in percent)

<table>
<thead>
<tr>
<th>Script</th>
<th>Bangla</th>
<th>Devnagari</th>
<th>English</th>
<th>Rejection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bangla</td>
<td>98.40</td>
<td>0.60</td>
<td>0.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Devnagari</td>
<td>7.15</td>
<td>89.73</td>
<td>3.12</td>
<td>0.00</td>
</tr>
<tr>
<td>English</td>
<td>0.40</td>
<td>0.00</td>
<td>99.20</td>
<td>0.40</td>
</tr>
<tr>
<td>Overall</td>
<td></td>
<td></td>
<td>94.69%</td>
<td></td>
</tr>
</tbody>
</table>

### Table 4: Line-wise script identification results

<table>
<thead>
<tr>
<th>Experimental Dataset</th>
<th>Bangla(%)</th>
<th>Devnagari(%)</th>
<th>English(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Handwritten lines</td>
<td>98.62</td>
<td>95.20</td>
<td>99.44</td>
</tr>
</tbody>
</table>

### Table 5: Comparison of word-wise script identification results (accuracy is given in percent)

<table>
<thead>
<tr>
<th>Method</th>
<th>Bangla</th>
<th>Devnagari</th>
<th>English</th>
<th>Overall Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roy and Majumdar[21]</td>
<td>96.77</td>
<td>97.68</td>
<td>93.61</td>
<td>96.02</td>
</tr>
<tr>
<td>Proposed method</td>
<td>98.53</td>
<td>92.04</td>
<td>99.13</td>
<td>96.57</td>
</tr>
</tbody>
</table>

**Results for line selection:** As discussed in Section 3.5.1, a filtering process was used to remove lines without date patterns. Depending on the value of the total count of date components, lines are eliminated for the next level of processing. It is noted that 83%, 84.39% and 64.6% of lines of Bangla, Devnagari and English scripts, respectively have been eliminated by setting the counter of the date element at 6. It is observed from the dataset that at least 6 date elements (i.e. 4 numerals and 2 punctuations) are required to represent a numeric date pattern. Fig. 8 shows the overall result of line selection according to the number of date components (elements). It shows that the number of lines decreases in each script as the number of date elements increases. It can be seen that 71, 88 and 65 lines of English, Bangla and Devnagari scripts, respectively, contain 10 or more date elements. It can be also seen that 439, 187 and 128 lines of English, Bangla and Devnagari scripts, respectively, contain at least 6 date elements. Lines containing 6 or more date elements are considered for date pattern searching.

**Results for date field extraction:** We have computed precision (P) and recall (R) measures to evaluate the quantitative performance of the date extraction.
system for these three scripts. Depending on the ground truth of the date, extracted sequences are considered to be a valid date sequence or not. The precision-recall measure for the components of the date fields are also extracted.

In this work, a DTW-based textual month identification approach is used. Precision-recall measures have been computed for handwritten Bangla, Devnagari, English (Roman) and printed English scripts. DTW-based method takes 2 seconds on average for month identification task. Fig. 10 shows precision-recall curves of month word identification for all the above scripts. A comparative study has also been performed on the printed English dataset to show that the DTW-based month word identification approach outperforms the SVM-based method proposed in [9] and the results are presented in Table 6. Here, precision and recall measures of DTW and SVM-based approaches have been presented. The DTW-based approach outperforms the SVM-based method in our experiments. The profile feature and the gradient feature have been used here for the DTW and SVM-based month word identification approach, respectively.

Table 6: Comparison of results for month word identification (Q1: Fields retrieved and relevant, Q2: Relevant fields in dataset, Q3: Fields retrieved).

<table>
<thead>
<tr>
<th>Method based on</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Precision (Q1/Q3*100)</th>
<th>Recall (Q1/Q2*100)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTW</td>
<td>96</td>
<td>114</td>
<td>129</td>
<td>74.42</td>
<td>84.21</td>
</tr>
<tr>
<td>SVM</td>
<td>89</td>
<td>114</td>
<td>126</td>
<td>70.63</td>
<td>78.07</td>
</tr>
</tbody>
</table>

Table 7: Component wise precision-recall measure (P: Precision , R: Recall)

<table>
<thead>
<tr>
<th>Script</th>
<th>Month</th>
<th>Numerals</th>
<th>Puncuations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P</td>
<td>R</td>
<td>P</td>
</tr>
<tr>
<td>Bangla</td>
<td>70.97</td>
<td>86.27</td>
<td>88.02</td>
</tr>
<tr>
<td>Devnagari</td>
<td>68.75</td>
<td>78.57</td>
<td>85.07</td>
</tr>
<tr>
<td>English</td>
<td>80.21</td>
<td>85.56</td>
<td>83.97</td>
</tr>
</tbody>
</table>

The precision-recall measure of date field component (Month, Numerals and
Figure 9: Qualitative results of numeric and semi-numeric dates (a-h) English handwritten, (i) Bangla handwritten, (j) Devnagari handwritten, (k) English printed. Extracted date fields are marked with a red box.

Punctuations) identification has also been computed separately for each script. Rows 2-5 of Table 8 show the performance of the date retrieval technique on Bangla, Devnagari, English scripts and the overall result, respectively. To get an impression of the qualitative results, a few examples of date field extraction of these scripts are shown in Fig. 9 and a precision-recall curve is also presented for all the scripts in Fig. 11. We have tested the same date field extraction technique on English printed documents. 500 printed lines containing different formats (numeric and semi-numeric) of date samples are extracted from the documents of the 'Tobacco-800' dataset and used in our experiments. A higher accuracy than that achieved on handwritten documents has been attained in this experiment. 85.09% recall and 100% precision have been achieved from
Figure 10: Precision Vs Recall curves for Month field extraction. Bangla, Devanagari, English and English printed PR curves are shown here.

Table 8: Precision recall measure of date field extraction (FR: Field for recognition, Q1: Fields retrieved and relevant, Q2: Relevant fields in dataset, Q3: Fields retrieved)

<table>
<thead>
<tr>
<th>FR</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Precision (Q1/Q3*100)</th>
<th>Recall (Q1/Q2*100)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bangla</td>
<td>106</td>
<td>134</td>
<td>106</td>
<td>100</td>
<td>79.10</td>
</tr>
<tr>
<td>Devnagari</td>
<td>82</td>
<td>110</td>
<td>82</td>
<td>100</td>
<td>74.55</td>
</tr>
<tr>
<td>English</td>
<td>329</td>
<td>419</td>
<td>329</td>
<td>100</td>
<td>78.52</td>
</tr>
<tr>
<td>Overall</td>
<td>512</td>
<td>663</td>
<td>512</td>
<td>100</td>
<td>77.22</td>
</tr>
</tbody>
</table>

Figure 11: Precision vs Recall curves for date field extraction on Bangla, Devnagari, English and English printed scripts.

**Error Analysis:** It is noted that the errors, which have been generated are due to improper classification of textual months and non-months, and some errors we found due to incorrect touching digit segmentation and misclassification of digits. Fig. 12 shows some erroneous results.
5. Conclusions

An automated system for date field extraction from multi-script handwritten documents has been presented here using machine intelligence and pattern recognition approaches for the use of a robot towards automatic retrieval of related documents based on captured date information. A DTW-based method has been employed in the present study to extract textual month components from scripts. Gradient-based features and SVMs are used for the identification of other date components such as digit, punctuation and contraction (e.g. st, nd, rd, th). Finally, different date patterns are searched from the sub-sequence of labelled components. Overall, the results of the approach presented are very encouraging, particularly as this research represents the first work on date field extraction in handwritten multi-lingual documents. In order to improve the proposed system, there is some scope to decrease segmentation errors at the word and character level. HMM-based segmentation-free approaches would be able to handle these errors which would lead to better results. Moreover, improvements at the recognition level may be achieved by such techniques. Use of a single feature extraction technique and a classifier would assist in designing a better system in terms of simplicity. Finally, an extension on regular expressions for searching would be an interesting direction for future work. As we mentioned earlier this would foster further research to facilitate generic solutions in document analysis, with different applications in robotics and automation.

References


