UNIVERSITÉ FRANÇOIS - RABELAIS DE TOURS

École Doctorale MIPTIS

Laboratoire d'Informatique, Equipe Reconnaissance des Formes et Analyse d'Images.

THÈSE présentée par :

Tanmoy MONDAL

soutenue le : 18 décembre 2015

pour obtenir le grade de : Docteur de l'université François - Rabelais de Tours

Discipline/ Spécialité : Informatique

From Time Series Signal Matching to Word Spotting in Multilingual Historical Document Images

THÈSE DIRIGÉE PAR :

RAMEL Jean-Yves	Professeur, Université François Rabelais de Tours
RAGOT Nicolas	Maître de conférences, Université François Rabelais de Tours
PAL Umapada	Professeur, Indian Statistical Institute, Kolkata, India

RAPPORTEURS :

VINCENT Nicole	Professeur, Université Paris Descartes, Paris
EGLIN Véronique	Professeur, Laboratoire d'Informatique en Image et Systémes d'information

---

JURY :

GATOS Basilis	Researcher (Président du jury), Institute of Informatics and Telecommunications, Greece
VINCENT Nicole	Professeur, Université Paris Descartes, Paris
EGLIN Véronique	Professeur, Laboratoire d'Informatique en Image et Systémes d'information
RAMEL Jean-Yves	Professeur, Université François Rabelais de Tours
RAGOT Nicolas	Maître de conférences, Université François Rabelais de Tours
PAL Umapada	Professeur, Indian Statistical Institute, Kolkata, India

Abstract

This thesis begins by giving an overview of the domain of typical time series data and how it has been used in matching, classification and retrieval problems. Several traditional domains of application of time series data and some conventional technique of dataset generation are also explained. Among many application of time series signals, one of the main domain of application is word image matching/word spotting. Word spotting can be defined as *"localization of interested word in the dataset without actually interpreting the content"*. For word spotting purpose, we represent the word images as time series signal and then several sequence matching techniques have been applied for spotting a word inside document image. One of the typical approach in literature is to match these signals by classical Dynamic Time Warping (DTW). However there exists several other improved versions of DTW along with other robust sequence matching techniques. We did a comparative study of classical DTW technique and many of its improved modifications, as well as other sequence matching techniques in the context of word spotting. An experimental study on historical documents is performed to evaluate the behavior of DTW's variants and other sequence matching techniques.

After evaluating several sequence matching techniques and analyzing their pros and cons, we propose a robust method to perform word spotting in degraded hand written and printed document images. A new sequence matching technique, called as Flexible Sequence Matching (FSM) algorithm is introduced for this task of word spotting by analyzing the drawbacks and advantages of other sequence matching techniques. Along with the characteristics of multiple matching (many-to-one and one-to-many), FSM is also capable of skipping existing outliers or noisy elements, irrespective of it's position in the target signal. More precisely, in the domain of word spotting, FSM has the ability to retrieve complete words or words containing only a part of the query. Furthermore, due to it's adaptable skipping capability, FSM is also less sensible to local variations in the spelling of words, and also to local degradation effects within the word image. The multiple matching capability (many-to-one, one-to-many) of FSM helps it to deal with stretching effects of query and/or target images.

One of the main drawback of FSM is that it can't skip outliers from query sequences, which make FSM inferior in certain conditions when there is a presence of noise in target signals as well as query signals. So, we propose another new sequence matching algorithm called as Exemplary Sequence Cardinality (ESC). ESC is an extension of FSM. It has all the qualities as FSM, in addition, ESC has the ability to skip the elements from query. In case of word spotting application, the outliers skipping capability of ESC makes it less sensible to local variations in the spelling of words, and also to noise present in the query and/or in the target word images. Thanks to it's outliers skipping facility from query sequence, this technique gives more flexibility to user for choosing query images. This

facility helps users to evade from rigorous searching for perfect query.

As another contribution in this thesis, we propose a shape code based word-image matching technique for word retrieval in documents, written in Indian languages. Each query word image to be searched is represented by a sequence of shape codes that corresponds to primitives. Then an inexact string matching technique is applied for measuring the similarity between the codes generated from the query word image and each candidate word images, obtained from the document.

In brief, we have investigated and explored several dynamic programming based sequence matching techniques for word spotting and conventional sequence matching applications.

Keywords : Dynamic Time Warping, Sequence Matching, Word Spotting, Image Matching, HOG Feature, Longest Common Subsequence, Shape Coding, Minimal Variance Matching, Optimal Sequence Bijection.

Contents

List of Tables

List of Figures

Chapter 1

Time Series and it's Representation

Contents

Abstract

In this chapter, we give an overview of time series data in general and how it has been used in classification problems. The goal of this chapter is also to expand the readers appreciation for the ubiquity of time series data in addition to the traditional application domains (e.g., stock market data, electrocardiograms, weather data, etc.). After posturing the general overview of application of time series data, we discuss some conventional techniques of dataset generation, coming from various domain of applications. Finally, we conclude by describing one popular time series dataset archive, which would be used to explore the proposed novel sequence matching techniques, mentioned in the subsequent chapters of this thesis.

1.1 Definition of Time Series

Due to the wide spread use of modern information technologies, a large number of time series data can be accumulated. So, there is a basic and important need generated for querying and indexing this huge data. Time series matching have been a prevalent area of research in the domain of pattern matching, data mining, signal processing, medical studies, weather forecasting, financial market, meteorology and many other statistical data analysis domains [Dietrich et al., 2004], [Eads et al., 2002], [Jalba et al., 2005], [Keogh and Ratanamahatana, 2005], [Smolinski et al., 2008], [Niennattrakul and Ratanamahatana, 2007], [Übeyli, 2008], [Yu et al., 2007], [Xi et al., 2006]. Time series data can be available from every aspect of human life. In a layman view, time series data can be defined as any sequence of real values that is recorded during a regular time interval. Often, time series data is accumulated by measurements of things, recorded over regular intervals of time (e.g., every 15 minutes, every month, every year, etc.). However, any sorts of time intervals (regular or irregular), which is ranging from millisecond upwards, are acceptable for generating time series data. Literature survey reveals hundreds of algorithms, which have been introduced to classify, cluster, segment, and index time series data, like nearest neighbor classifier, various clustering techniques, support vector machine; neural networks [Eads et al., 2002], [Güler and Ubeyli, 2005], [Ratanamahatana and Keogh, 2004b] etc. Whereas classical algorithms assumes relatively low sequence dimensionality, time series data mining algorithms must be able to deal with dimensionality in the hundreds or thousands. Except the issues of high computational time, other problems exist for high dimensional time series sequence matching. One of the critical issue is that as dimensionality increases, all objects become essentially equidistant to each other, and thus classification and clustering lose their meaning. This surprising characteristic is known as the curse of dimensionality and has been the subject of extensive research. Also, presence of noise and intrinsic structure of time series make them difficult to process and algorithms need to be adapted.

1.2 Introduction

Due to the growing utility from different field of applications, a notable research effort has been devoted in time series matching. In this section, we give an brief overview of case studies of the related works in domain of time series, more specifically in time series matching, retrieval and classification. The classic and efficient way to measure the distance between time series signals is by Euclidean metric [Faloutsos et al., 1994]. The simplicity of Euclidean distance [Faloutsos et al., 1994], attracts the attention of research community and makes it most popular dissimilarity measure in time series data mining [Agrawal et al., 1993] [Keogh et al., 2001]. It requires that both input sequences be of the same length, and but it is sensitive to distortions and shifting along the time axis [Ratanamahatana and Keogh, 2005] [Jagadish and Faloutsos, 1998]. There are several other techniques to classify time series signal are mentioned in literature. Through the application of recurrent neural networks, Husken et.al. [Hüsken and Stagge, 2003] proposed a technique for time series classification. For the classification of electroencephalogram (EEG) signals, Guler et al. [Güler and Ubeyli, 2005] presented a wavelet based adaptive neuro-fuzzy interference system model. For word image matching, Rath and Manmatha

[Rath and Manmatha, 2006] used DTW and compared the performance of their system with some conventional techniques, including affine-correlated Euclidean distance mapping, the shape context algorithm and correlation using sum of squared differences. For time series classification, a time series representation model known as, Derivative Time Series Segment Approximation(DSA) [Gullo et al., 2009] was proposed. This model combines the notion for derivative estimation, segmentation and segment approximation, for supporting accurate and fast similarity detection in time series data. A hybrid classification algorithm, employing evolutionary computation for feature extraction and support vector machine for classification with the selected features is proposed by Eads et.al. [Eads et al., 2002]. A semi supervised technique for building classifiers is proposed by Wei et. al. [Wei and Keogh, 2006]. One of the crucial point in time series classification is how to measure the dissimilarity of time series. A comparative, well documented overview can be found in [Ding et al., 2008]. Another direction of research with time series data is of predicting future values from a series of past data extending up to the present. The primary goal time series prediction algorithms is to improve prediction accuracy. There have been many techniques to achieve this goal, for detail discussion, please see [Wakuya and Shida, 2002]. In the following section, we mainly focus our discussion on time series classification and matching.

One of the most fundamental tasks in data mining domain is time series classification. The classification of sequence of data is primarily performed by similarity matching between sequences. There are several real life examples of the applications of time series classifications, e.g. image and pattern recognition, spam filtering, genome diagnosis, and text mining of large amount of documents etc. The main goal of classification is to map input data into predefined groups. The classifications are generally performed based on priorly defined groups, and the tasks of any classification algorithm is to correctly classify any data in it's proper class. The classification techniques can be broadly categorized into two sub category: *i)* learning based and *ii)* learning free. The learning based techniques can further be grouped into two groups, called as *a)* supervised learning *b)* unsupervised learning. The usual process of supervised learning is followed by predefining data for training process and learning is made to recognize patterns of interest, whereas in case unsupervised learning, the task is to find hidden structure in unlabeled data. The examples used for learning are always unlabeled, so there is no error or reward signal which can be repetitively used to learn the underlying algorithm.

In case of learning free approaches, the task of unclassified data classification, are often performed by several pattern recognition techniques. Pattern recognition can be defined as the classification technique where an input pattern is classified into one of several predefined classes based on the similarity to these predefined classes. The accuracy of classification algorithms is calculated by determining the percentage of objects identified as the correct class. Among many other classification techniques, two of the popular approach of time series classification are *Decision trees* and *Nearest Neighbor classifier*. In the case of decision tree, a set of rules are generated from training data and based on these rules, a new data is classified. One of the primary bottleneck of raw time series data is it's high dimensionality and the existing noise into it. These impediment of raw time series data results in deep, bushy tree. One convenient approach is to represent the time series as regression tree and then to use it as decision tree training. For the case of time series matching approaches,

the classification accuracy is measured by running a one-nearest neighbor or k nearest neighbors algorithm using leaving-one-out method. In each iteration, one most dissimilar item is excluded from the dataset, this process is repeated n times, where n is the total number of items in the dataset, until the calculation of closest match (for one nearest neighbor). In the case of k nearest neighbors, the same process is performed for finding k closest matches. Finally, based on the distance value between the query object and other close matches, one label is assigned to the query object. The classification accuracy is the percentage of objects that are correctly classified.

Recent advances in technology has made computers more powerful, which inherently motivated researchers in the domain of time series indexing or retrieval [Agrawal et al., 1993]. The tasks of time series indexing and query by content can be divided into two broad categories: i) whole matching and ii) subsequence matching.

- **Whole Matching**: when time series query is matched against a database of individual time series of the same length, to identify the ones closer to the query sequence.
- **Subsequence Matching**: when the query sequence is shorter than target sequence. The matching could be done by sliding along the longer sequence or directly by finding the best sub-part into the longer sequence, which optimally corresponds the query sequence.

While there are literally hundreds of methods proposed for whole sequence matching (see e.g., [Keogh and Kasetty, 2003] and references therein), in practice, its application is limited to cases where some information about the data is known a priori. Subsequence matching can be thought as special kind of whole sequence matching technique, where whole sequence matching is attained by dividing the long sequence into equal size (same as the length of query), small sequences. This division can be performed based on specific period or, more arbitrarily by its shape. For example, a long speech signal can be divided into individual signals of phonemes and can be matched with a query phoneme. This informal idea has been used by many researchers. Given a database of sequences, the general way to find the closest match to a given query sequence Q, is to perform a linear or sequential scan of series by series. Each sequence is retrieved from disk and its distance to the query Q is calculated according to the pre-decided distance measure to calculate the closest matched sequence in the database. This brute-force technique is costly to implement due to its multiple accesses to the disk. To alleviate this problem, one lower bound of the distance can be determined and based in this lower bound, some irrelevant sequences in database can be pruned off, which in terms help to reduce some unnecessary costlier disk accesses. The lower bounding distance of a particular time series pair indicates that they are impossible to be best matches.

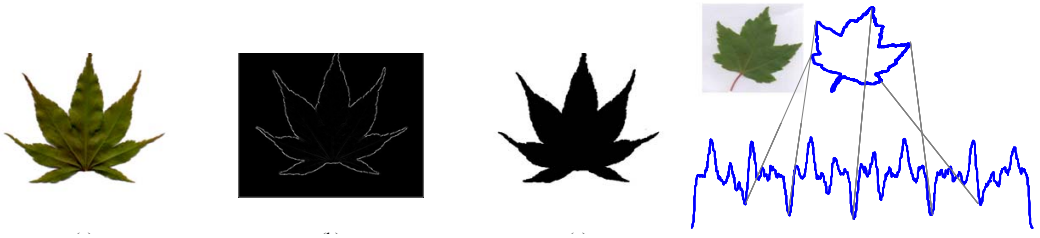
1.3 Application of Time Series in Various Domain

Time series data can be extracted from almost every aspect of human life. A time series database (or sequence database can be defined as a sequences of ordered events, with or without concrete notions of time. Based on this definition, some of the multimedia data

or the less-intuitive domains can be transformed into one or two dimensional time series data.

1.3.1 Image Retrieval

The task of image retrieval is a big stake holder in the domain of information retrieval [Ratanamahatana, 2005]. With the recent advancement of digitization systems, a large and distributed collections of scientific, artistic, technical, and commercial images have become more prevalent, and easily accessible. Thus the demand for more sophisticated and precise methods to perform similarity or semantic based queries have increased. The image retrieval techniques can be divided into two broad categories i) annotation based ii) content based. There are several issues with annotation based image retrieval system e.g. labor dependency and lack of consistency due to large amount of manual annotations required, whereas the later one gives an intuitive query interface and satisfactory retrieval performance. As an example of leaf image retrieval system, the leaf shape can be represented by using a distance curve, which is a sequence of distances between its center point and every point along the leaf contour. One good distinguishing factor for leaf image



(a) (b)

Figure 1.1: (a) Image preprocessing steps: Original image, edges by Canny edge detection algorithm and image binarization (left to right)[1]. (b) Process of converting the image into a pseudo time series[3].

retrieval system is blade (the leaf structures) based matching. Most plants has an unique shape of leafs that consists of one or more number of blades (see [Tak, 2008]). By generating distance curve from the leaf image, information of blade like distinguishing factors can be easily represented. The process for obtaining the distance curve is quite straight forward. One simple way to generate contours from the leaf image is to, binarize it first and then to apply some edge detection technique e.g. canny edge detector (see Fig. 1.1a). After obtaining the boundary of the image, the geometric center point can be calculated by averaging the X and Y coordinates of all the pixels, inside the boundary of the leaf. The relative position of this point does not change during scale and rotation. The time series sequence can be generated by calculating the distance between all the points along the contour in a clock -wise direction from the center point (see Fig. 1.1b). The curve generated from two different size image of same leaf can have different magnitude. To

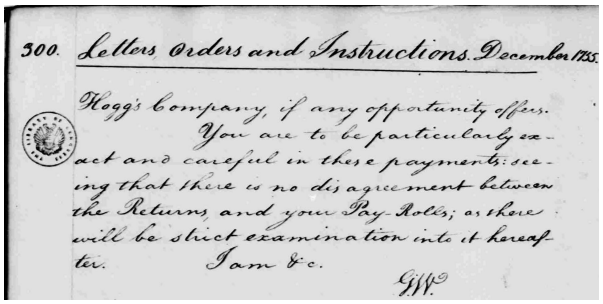
[1]Reprinted with permission from Chotirat Ratanamahatana, Improving Efficiency and Effectiveness of Dynamic Time Warping in Large Time Series Databases, PhD thesis at University of California Riverside, 2005.

avoid these sorts of problems, scaling invariant sequence matching technique should be considered. One simple process of rescaling all present sequences in the database is by calculating the maximum magnitude (m) from all the sequences in the database. Now any sequence (s) can be rescaled by multiplying all of it's elements by ($\frac{m}{n}$), where n is the maximum magnitude of the particular sequence.

Hand written and machine printed word Images matching and retrieval is a well known sub domain in the domain of image retrieval. With the recent advancement of digitization technology, there have been many historical manuscripts are digitized by various public libraries and several public and private organizations. Although these documents are digitized, indexing and searching some contextual information inside these huge collection of documents is a troublesome task. But, accessing information from these knowledge resources are important for researchers/biographers. Recognition and retrieval of handwriting is a challenging task. Compared to the recognition of on-line handwriting, off-line handwriting recognition is more difficult [Ratanamahatana, 2005]. Moreover, the problem of transcribing and indexing handwritten and printed documents is more troublesome. The high degradation effects, font and script variabilities, presence of noise are the factors, which makes the recognition difficult on historical documents. Compared to on-line hand writing problem, the problem of indexing historical archives is difficult. In case of on-line handwriting recognition, the information about pen acceleration, time stamp, pen pressure information could be used for the recognition purpose. But for the case of off-line hand writing recognition, there are no such information available and the hand writing on a page has to be treated as an image. Moreover, in the case of historical hand written documents, there are often highly stylized scripts and writing by multiple writers. For example, Fig. 1.2 shows an example of hand written text by George Washington. As it is visible from the given image that even for modern readers with little knowledge on cursive hand writing, it is difficult to understand the contents. Instead of recognizing complete text lines, a new direction of research work has been emerged, which is known as: *word spotting*. Word spotting can be defined as the technique of recognizing any word image, without transcribing text contents of image. Among several techniques, one way to do this is to represent word images as a time series signals (see Fig. 1.2) and then applying time series matching algorithms for matching two signals extracted from two different word images. Based on the calculated distance between query and target word images, these ones are ranked and hence retrieved.

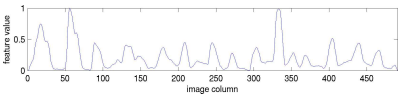
1.3.2 Acoustic Wildlife Management

The interaction techniques between animals has always been a mystery for man-kind. There have been several attempts by human to understand the science behind these interaction techniques. To discern the complex intercommunication techniques, biologist need to be able to track their locations. Acoustic wildlife management is a domain of study to measure the health of an ecosystem, e.g. monitoring the chirping of birds, croaking of frog, hisses of snakes, screeching of bats and other insects, by adaptive, embedded networked sensing technologies [Trifa et al., 2007]. An important application of adaptive sensory systems for biology is the surveillance of natural systems with the purpose of describing their structure and behavior. To avoid the presence of human observers in the field, which is

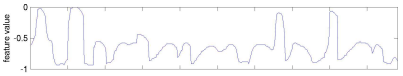


(a) Sample hand written page

(b) Sample key-word *Alexandria*



(c) Normalized projection profile



(d) Normalized upper word profile

Figure 1.2: (a) An example of handwritten text from George Washington dataset[3]. (b) One example of a key-word *Alexandria*, after removing the existing slant from the writing. There are many techniques available to convert handwriting into time series: (c) the projection profile i.e. total number of foreground pixels in each column. (d) the upper word profile i.e. the location of top foreground pixels in each column.

both time consuming and troublesome to the habitat under observation, adaptive sensory systems are highly useful. The signal shown in Fig. 1.3, is the screech of Strigiformes

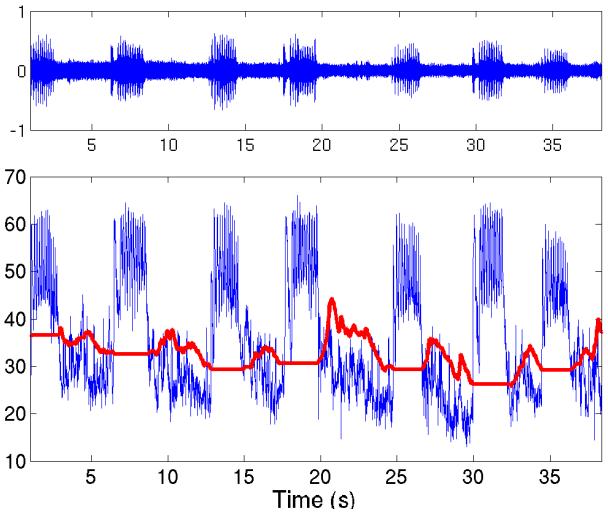


Figure 1.3: Adaptive detection algorithm. Top: Amplitude of the signal of a field recording. Bottom: evolution of the energy in the desired signal represented by thin line. The thick line represents the detection threshold and if the amplitude of any signal is greater than this threshod value, it is considered as valid signal.

(owls). The domain experts have noted that most owls repeat their calls in a window of eight to ten seconds and that the calls last from one to three seconds [Vincenzo, 2002]. But an important drawback of such sensory system is that these systems are typically installed to monitor twenty four hours a day, so the memory limits for storage or bandwidth limits

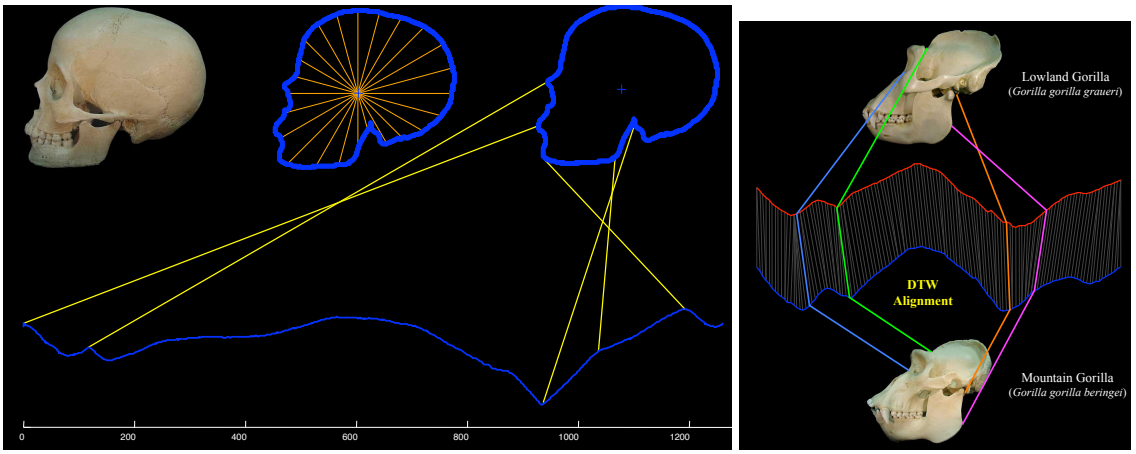
for transmission, form a bottleneck on how much data can be retained in field deployed sensors. For example, by using a simple thresholding algorithm, the researchers was able to reduce half an hour of raw recording to only 13 seconds of useful audio [Trifa et al., 2007]. One kind of strange rhythm/reiteration can be observed in the singing or calls of animals of many species for intra-specific communication [Dawson and Efford, 2009]. So by applying an intelligent algorithm, the amount of data retained can be reduced by identifying the repetitive nature of certain bird calls, while it can also reduce the false positive rate. For example the sound data, captured from an installed sensor in a field, shows the following behavior, mentioned in Fig.1.3. It is visible that these kinds of signal has a specific cyclic pattern, which is a distinguishable property of such signals. Two or multiple different captured signals can be matched or classified by popular sequence matching techniques (e.g. DTW), which could provide many information e.g. the type of animals, the different species of a particular animal etc. Moreover, if these signals can be properly analyzed, matched and classified, it can help to understand intercommunication and interaction techniques of animals, birds, insects etc.

1.3.3 Shape Matching

Matching two dimensional shapes is an important problem with applications in the domain of biometrics, industry, medicine and defense etc [Ratanamahatana, 2005]. The problem of shape matching can be defined as the problem of describing a shape and calculating it's similarity with others. The distance measure used for the shape matching, must be invariant to many distortions, including scale, offset, noise, partial occlusion, etc. Although most of these distortions are relatively easy to handle, and particularly if the well-known technique is used for converting the shapes into time series. There are several techniques available in the literature for converting shapes into time series [Adamek and O'Connor, 2004], [Attalla and Siy, 2005], [Antonio Cardone, Ra K. Gupta, 2003].

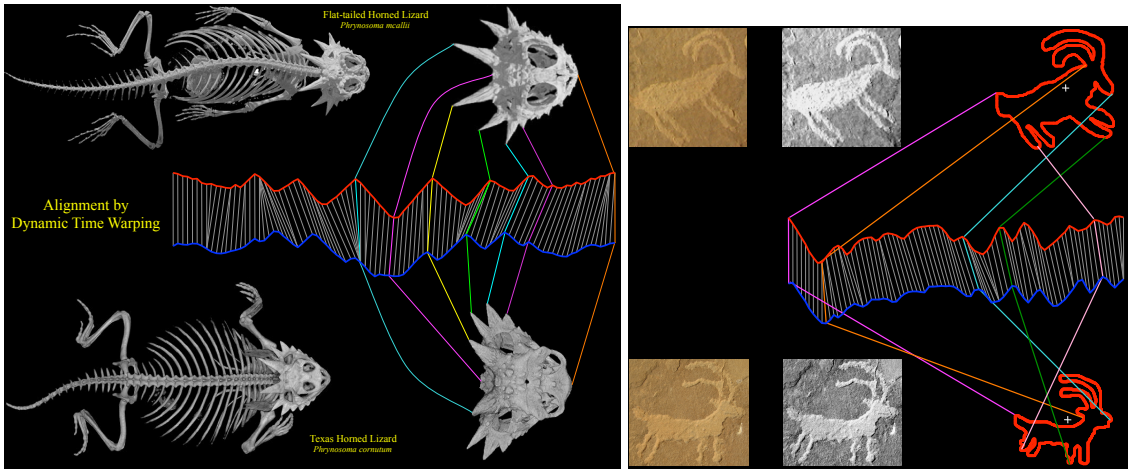
1.3.4 Text Mining

There have been several technique mentioned in the literature to transform text data into a time series representation [Ratanamahatana, 2005]. One of the known technique to transform is to utilize different combination of granularity (i.e. character or word level) to extract text units from strings. For example, unigram, bigram or trigram could be used to form subsequence of text line [Yang and Lee,]. The work mentioned in [Yang and Lee,] transform texts into time series representation in the case when translated text is available in both English and Spanish. The basic idea is to convert the bible text into bit streams based on the occurrences of a particular word in the text. Then a time series can be generated based on the number of word occurrences within a predefined sliding window across the bit streams. As an example, let's consider a sentence : *In the beginning God created the heaven and the earth* and let's focus on the word *God*. The mentioned sentence can be represented by "0001000000" by considering the word "God" as a hit. But the generation of bit stream depends on the size of sliding window. For example, if we consider that the size of sliding window is 3 then the running example will generate the bit stream as: 01110000. The idea behind this approach is based on the assumption that for each word



(a) (b)

Figure 1.4: (a)The process of converting shapes into time series: Initially from the raw bitmap image of human skull (left most one), a contour is detected. After that the distance from every point on the contour profile to the center is measured (middle one) and treated as the Y-axis of a time series of length n (right most one). (b) The skull of Lowland Gorilla and Mountain Gorilla are morphologically similar, except a small variability in different proportions. DTW alike time series matching techniques can be used to align homologous features in the time series representation space.



(a) (b)

Figure 1.5: (a) The time series signal extracted from Flat-tailed Horned Lizard (Phrynosoma mcallii) (top one). Texas Horned Lizard (Phrynosoma cornutum, bottom one) and matched by DTW. (b) The images of two similar look alike fossils and the time series signals extracted from these historical wall painting are matched by DTW.

in English, there must be a corresponding word in Spanish and this word in Spanish also occur in the same vicinity of it's corresponding English word (see Fig. 1.6). For example, to find the similarity between two same documents, written in different languages (e.g. Bible

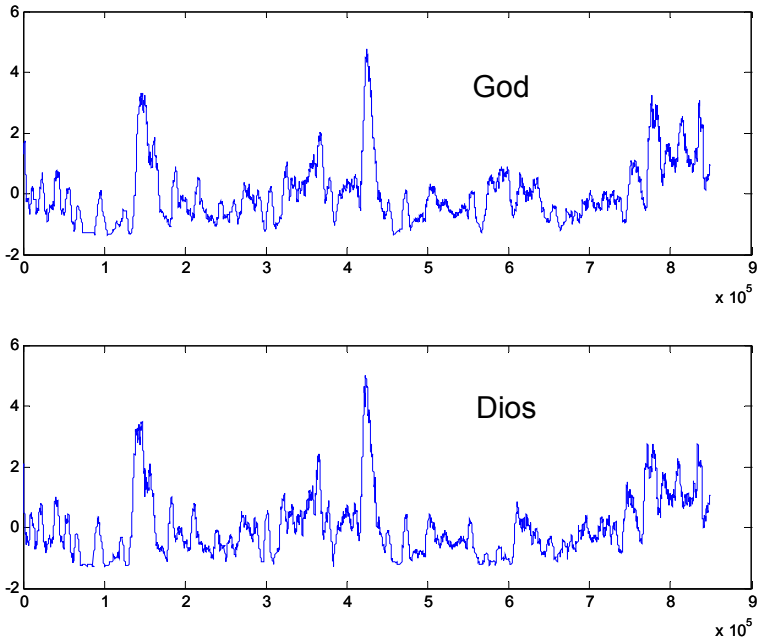


Figure 1.6: By using a window size of 6000 words, the times series of the number of occurrences of the word 'God' in English (top) and 'Dios' in Spanish (bottom) in bible text (z-normalized and reinterpolated to the same length) are shown. It can be visible that when the window size is quite large, the two time series are almost identical[3].

in English and Spanish), the frequency of occurrences of a particular word (or multiple words) and it's position in the text can be mapped as a time series signals, which then can be compared by any relevant time series matching techniques (e.g. DTW). The small discrepancy in total number of words in entire text and the position of words within the sentence between two languages due to different nature of language structure should be handled by concerned time series matching algorithms.

1.3.5 Genome/DNA Mining

To evaluate evolutionary relationship between different organisms, the understanding of the differences between DNA symbolic records are required. To achieve this, global multiple sequence alignment is needed. But performing perfect similarity measure is a difficult task and that's why this process requires user intervention for exact alignment of the sequences. To characterize the similar sequences, the existing similarity between shapes of the curves has to be characterized. The common approach to represent DNA sequence by symbolic notation (e.g. symbols A,C,G,T for DNA bases). But the problem of these methods are that they are highly computationally expensive. Moreover these techniques are highly sensitive to errors and mathematical evaluation is difficult with these kinds of representations. One of the principal step in genome matching problem is to transform DNA symbolic record into numerical form. The process of transformation is very important because all biological properties described in the original symbolical form must be retained

in the final genomic signal [Yau et al., 2003] [Cristea, 2002]. The curves for cumulated phase for human, rhesus macaque and chicken are shown in Fig. 1.7. It can be visible that there are high similarity between human and rhesus genomic signals whereas the curve of chicken is highly different from all other chosen organisms because all other species are mammals. However certain degree of similarity is visible between the curves of mammals and chicken.

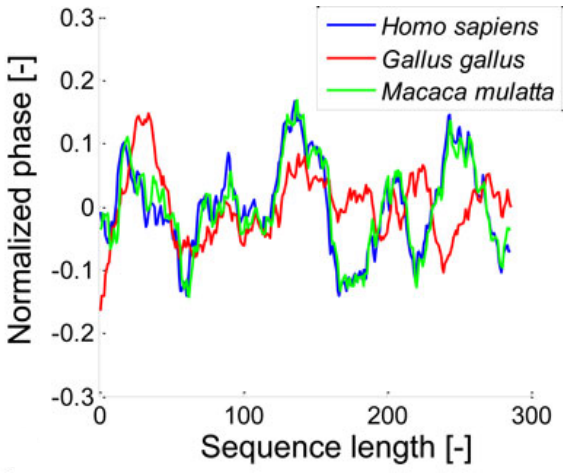


Figure 1.7: The record of cumulated phase of the DNA sequences of three different organisms.

1.3.6 Phoneme Classification

Automatic classification of phonemes are one of the highly researched topic in the domain of speech recognition. Phonemes are the smallest units of intelligible sound, produced by human, whereas phonetic spelling is the sequence of phonemes that a word comprises. For example, the word *boss* has two phonetic spelling for British and American accents. Two pronunciation of the word *boss* are shown in following Fig. 1.8. In earlier decade, the phonetic segmentation was a big challenge but with the recent advancement of various phonetic segmentation techniques, automatic generation of million phonemes has become more easier. Automatic speech recognition is highly useful for robust speech recognition, speech quality scoring and dialect/accent recognition. Classifying these phonemes is a challenging task for data mining community because of large number of classes and the complexities of existing algorithms. For example, there are almost 107 phonemes, 52 diacritics and 4 prosodic marks (covering various languages [2]) according to *International Phonetic Alphabets (IPA)*. The other issue is that the basic units of phonemes is highly depended on speakers, dialects, accents, noise in the environment and errors in automatic generation of phonemes. Most of the works in last 20 years in this domain are based on the well known speech TIMIT dataset [Garofolo et al., 1993]. TIMIT is specifically designed for speaker invariant phoneme classification. The main tasks for automated phonotactic

[2] `https://en.wikipedia.org/wiki/International_Phonetic_Alphabet`

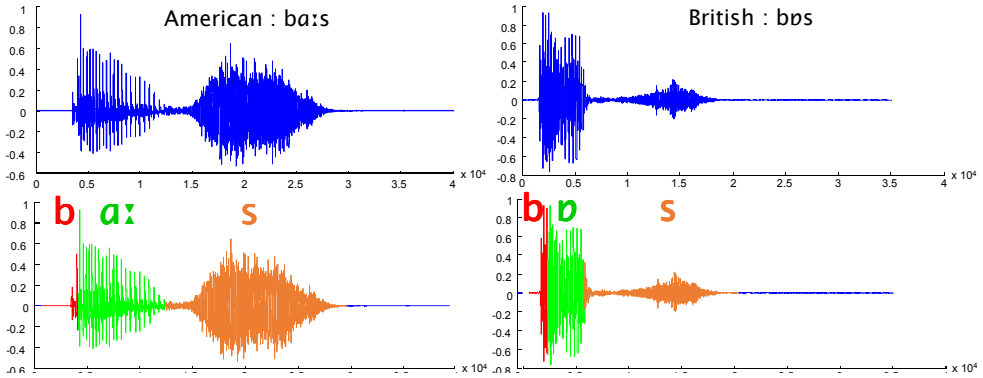


Figure 1.8: Two waveforms generated from the pronunciation of the word boss by American and British accented speakers. It is visible from the signal that American accent has a prolonged vowel, while the British accent does not have the elongated vowel. Also the British accent places less stress on the endings. The perfect segmentation of the phonemes produced by Forced Aligner [Jiahong Yuan, 2008] is shown in color.

processing [Rama, 2013] is phoneme segmentation, classification, and recognition. Segmentation finds the phoneme boundaries inside a speech sequence. Classification identifies each individual phoneme, and recognition decodes the sequence of phonemes taking into account segmentation and classification errors. These three tasks are sequential but interdependent. Various time series matching techniques (e.g DTW) could be used for performing these aforementioned tasks in pipeline or individually.

1.4 Time Series Datasets

To explore the spectrum of research in time series domain, and especially to evaluate classification and matching accuracy of the techniques, proposed by researchers, one well know bench mark of time series data have been created. This benchmark dataset is known as UCR dataset[3]. This time series dataset has been highly used by several researchers, working in the domain of time series matching and classification problem. UCR archive consists of total 47 datasets (see Table 1.2). These data sets include real-life time series, synthetic time series, and generic time series, come from different application domains and are obtained from the UCR Time Series Classification/Clustering archive ([Chen et al., 2105]). Information on the data sets used is given in Table 1.2. For the detailed descriptions of the data sets, please see [Chen et al., 2105]. In the following section, we describe in more details, some of these datasets.

1.4.1 Trace Dataset

This synthetic dataset was designed to simulate instrumentation failures in a nuclear power plant (Transient Classification Benchmark (Trace Project)) [Ratanamahatana, 2005] [Roverso, 2000]. The original dataset consists of 16 classes and 50 instances in each

[3] `http://www.cs.ucr.edu/~eamonn/time_series_data/`

class. Four features are included from each instances. For reducing the complexity of the experiments, only second feature of classes 2 and 6, and the third feature of classes 3 and 7 are extracted in the UCR dataset. Hence, this modification results in a 4-class problem, where in the reduced set, the training and testing dataset is consisting of 200 instances, 50 for each class. All instances are interpolated to have the same length of 275 data points.

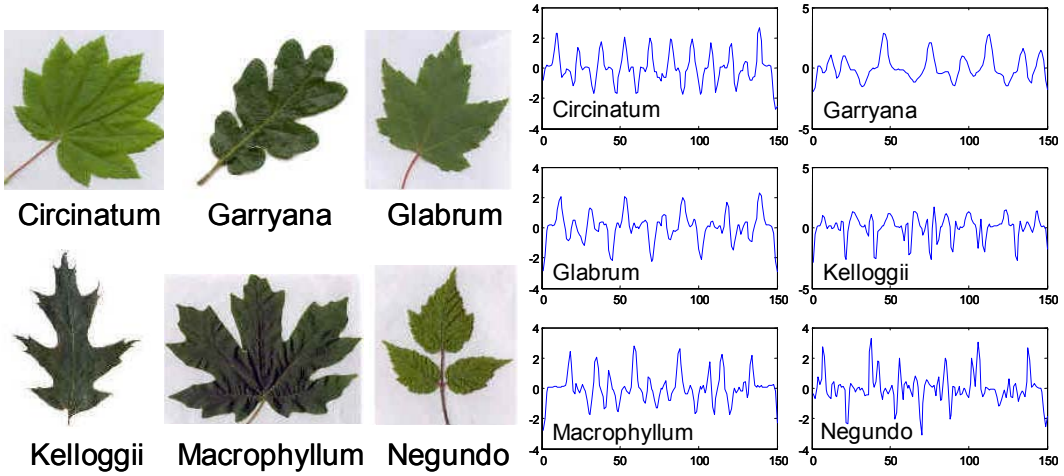


Figure 1.9: The record of cumulated phase of the DNA sequences of three different organisms[3].

1.4.2 Leaf Dataset

Six different species of leaf images are considered in this dataset of leaf images. This dataset was created by Ratanamahatana et.al. [Ratanamahatana, 2005]. The original isolated leaf color images are obtained from the Machine Learning Group, Oregon State University [4]. The dataset contains four species of Maple plant and two different species of Oak plants. The following Table. 1.1 describes the details about this dataset. The illus-

Species	**No. of images**	**Class**
Circinatum (Vine Maple)	66	1
Garryana (Oregon White Oak)	84	2
Glabrum (Douglasii Maple)	75	3
Kelloggii (California Black Oak)	97	4
Macrophyllum (Big Leaf Maple)	82	5
Negundo (Boxelder Maple)	38	6
Total	442	

Table 1.1: Details of leaf dataset

[4] `http://web.engr.oregonstate.edu/~tgd/leaves/dataset/isolated.tar`

tration of each species of leaf images can be visible in Fig.1.9a. The same aforementioned approach (see Section 1.3.1) for converting leaf image into time series signal is used here also. The rotation and size of leaf images are not relevant because the time series is always generated from the beginning of the stem or bottom of the leaf if no stem is present in the leaf. The extracted time series are z-normalized and linearly interpolated to have the same length of 150 data points (see Fig.1.9b).

1.4.3 Face Dataset

To explore the utility of the time series matching techniques on face classification problem, another dataset was created based on the head profile taken from photographs of four individuals [Ratanamahatana, 2005]. To generate 112 instances of this dataset, each person was instructed to show different expressions on the face, such as smiling, talking, frowning, etc. A similar aforementioned technique of transforming an image into time series is

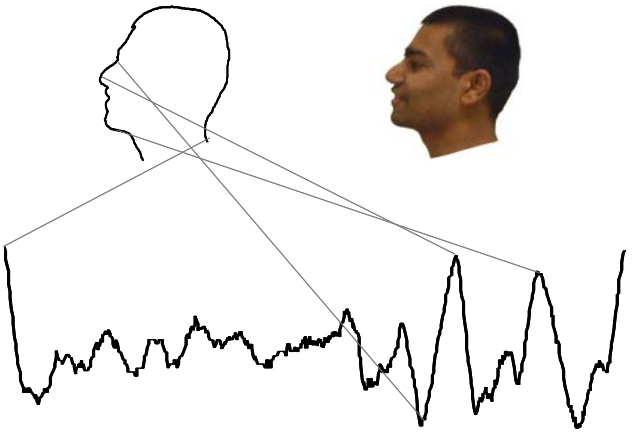


Figure 1.10: Starting from the neck area, the head profile is unrolled into a z-normalized and linearly interpolated *"pseudo time series"* [3].

applied here; starting from the neck area, the head profile is unrolled into a z-normalized and linearly interpolated *"time series"* (having 350 data points), as shown in Fig. 1.10.

1.5 Conclusion

In this chapter, we have presented the importance of time series for the research in various domain of applications. From the discussion in this chapter, we have seen that there are many field of application, each of them has it's own specificities. It is explained in this chapter that among various utilities of time series signals, classification, matching and retrieval tools are of huge importance. We have also seen that there exists some applications, which are not originally from the domain of time series but there are well established techniques to generate time series signals from these domains and then to take benefits of existing tools of the time series domain e.g. image matching, DNA sequence matching, text mining etc. In the following chapters, we will focus on one specific application domain

Table 1.2: Statistics of UCR datasets.

Data set	Number of classes	Size of training set	Size of testing set	Time series length
50Words	50	450	455	270
Adiac	37	390	391	176
Beef	5	30	30	470
Car	4	60	60	577
CBF	3	30	900	128
ChlorineConcentration	3	467	3840	166
CinC_ECG_torso	4	40	1380	1639
Coffee	2	28	28	286
Cricket_X	12	390	390	300
Cricket_Y	12	390	390	300
Cricket_Z	12	390	390	300
DiatomSizeReduction	4	16	306	345
ECG200	2	100	100	96
ECGFiveDays	2	23	861	136
Face (all)	14	560	1690	131
Face (four)	4	24	88	350
FacesUCR	14	200	2050	131
Fish	7	175	175	463
Gun-Point	2	50	150	150
Haptics	5	155	308	1092
InlineSkate	7	100	550	1882
ItalyPowerDemand	2	67	1029	24
Lightning-2	2	60	61	637
Lightning-7	7	70	73	319
MALLAT	8	55	2345	1024
MedicalImages	10	381	760	99
MoteStrain	2	20	1252	84
Non-Invasive Thorax1	42	1800	1965	750
Non-Invasive Thorax2	42	1800	1965	750
OliveOil	4	30	30	570
OSU Leaf	6	200	242	427
Plane	7	105	105	144
SonyAIBORobot Surface	2	20	601	70
SonyAIBORobot SurfaceII	2	27	953	65
StarLightCurves	3	1000	8236	1024
Swedish Leaf	15	500	625	128
Symbols	6	25	995	398
Synthetic Control	6	300	300	60
Trace	4	100	100	275
Two Patterns	4	1000	4000	128
TwoLeadECG	2	23	1139	82
uWaveGestureLibrary_X	8	896	3582	315
uWaveGestureLibrary_Y	8	896	3582	315
uWaveGestureLibrary_Z	8	896	3582	315
Wafer	2	1000	6174	152
WordsSynonyms	25	267	638	270
Yoga	2	300	3000	426

to show the interest of the approach, we propose to enhance classical sequence matching techniques.

Chapter 2

Time Series for Word Spotting

Contents

Abstract

As seen in the previous chapter time series signals can be extracted from almost every aspect of human life and there are many applications of time series matching techniques such as matching, retrieval, classification etc. Time series signal matching is a prevalent area of research for the data mining community. One of interesting application is word image matching. In this chapter, we will only focus on one specific research direction, known as *word spotting*. Word spotting can be defined as *"localization of interested word in the dataset without actually interpreting the content"*. In this chapter, we explain the process of representing word images as time series signal and then how the several sequence matching techniques have been applied for spotting a word inside document image. Along with it, we present an literature review of relevant word spotting techniques. We also present a general word spotting architecture, used throughout this thesis. We also have presented a fast word image retrieval technique using *Kernelized Sensitive Hashing*. Finally, the descriptions of all the datasets, used in this thesis are given in this chapter.

2.1 From Time Series Matching to Word Spotting

In the previous chapter, we have seen that the time series data is available in almost every aspect of human life. There are plenty of application domains of time series signal matching. Among many domain of applications, one useful application of time series matching is image matching. Images is nothing but a 2D matrix, where the values in the matrix can vary from 0-255. There have been hundreds of techniques available for extracting significant features, which can represents the distinguishing characteristics of images. Among these ones, some could be considered as a sequence 2D features. For example, image representation technique mentioned in Section 1.3.1 of Chapter 1, where structure of a leaf image is represented in terms of time series signal. Another way of image representation could be image column based representation, where from each column of image, we can extract features like average intensity of all pixels in the column, number of foreground and background pixels in the column (on binarized images) etc. Then these obtained sequences, considered from left to right for example, can be matched for finding the similarity between two images, by using time series matching algorithms. This is typically, what can be done for searching words inside document images, without recognizing them explicitly, i.e. word spotting.

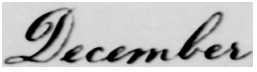
2.1.1 Introduction to Word Spotting

Today world of digitization, has shown a stirring alternative to preserve and provide an easy access to precious ancient manuscripts. High quality document digitization has opened up a new way for historians and research scholars for easy and hassle free access to these documents. Retrieving information from these invaluable knowledge resources are highly useful for interpreting and understanding history of various domains and to know cultural and societal heritage. Big software giants like Google and Yahoo have shown their high interest to avail the scanned ancient printed and handwritten manuscripts through their respective search engines. But only digitization can't be much helpful and can't meet the user requirement completely, if these huge collection of manuscripts can't be indexed and made search-able. The performances of available OCR engines of different separate scripts and languages for printed documents gives excellent recognition accuracy for the case of contemporary good quality documents. But these OCR engines drastically fail to perform in the case of old historical manuscripts. For the case of historical degraded cursive handwritten off-line document images, particular domain specific success [Plamondon and Srihari, 2000, Wang et al., 2014c, Pal et al., 2012, Senior and Robinson, 1998, Günter and Bunke, 2005] has been achieved in the direction of character recognition and overall transcription of the complete documents. But an expensive process of learning is attached with most of these notable off-line handwriting recognition techniques. The *"writing and font style variability"*, *"linguistics and script dependencies"* and *"document's poor quality caused by high degradations effects"* are the critical bottlenecks of such systems. Moreover, the process of manual or semi-automatic transcription of the entire text of a handwritten or printed documents for searching any particular word is a tedious and laborious job. For these reasons research has been emphasized on word spotting in handwritten manuscripts and historical printed document images. Word spotting is a relatively new alternative for

information retrieval in ancient manuscripts. In the following section, an extensive literature reviews on word spotting is mentioned. It can be visible from these reviews that, there are still some important unresolved problems in this domain. For example, spotting of the words independent of the variation of scripts and languages has not been properly addressed by the research community. Much work has not been done to propose a robust word spotting technique for handling noise and degradation effects present in the historical document. In most of the languages, there exists several variation of some particular words. For example, the French word *cheval (horse)* can have derivatives like *"chevalerie"*, *"chevaux"*, *"chevalier"*. In old French, other derivatives also exists due to lexical variations: *"chevallerie"*,*"chevaus"*; and also the *"v"* is often printed as *"u"*. So, a same word can be written in different ways and retrieving similar words can be interest for the user. Unfortunately, much research works has not been addressed in the direction of taking into account the word derivatives.

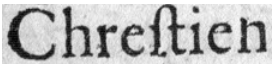
2.1.2 Principles of Word Spotting

In this chapter, we focus on word spotting in handwritten and printed documents, such as letters, memorandum, or manuscripts. Without transcribing the data, a user should be able to search for any word, just like using a search engine and the result of such search may look like as the one shown in Fig. 2.1. The spotting system just returns the likelihood or similarity between query word and accuracies in the images. If the likelihood of a target occurring in the image is above a given threshold, then this target is returned as a positive match along with it's position in the image. In case of multiple presence of target words are considered, the word spotting system would return the ranks of these occurrences, based on their likelihood/similarity measure with the query word. The following Fig. 2.1 illustrates a layman view of desirable word spotting outcome of the system, where one positive match in the image are marked by a rectangle box by considering 3 different query words and 3 different datasets respectively.

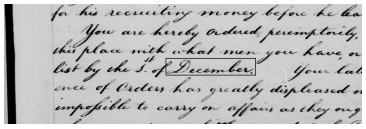


(a) Query word from GW dataset

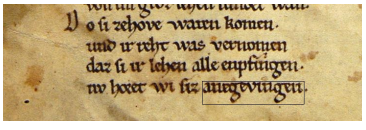
(b) Query word from Parzival dataset



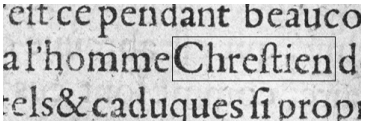
(c) Query word from CESR dataset



(d) Sample page of GW dataset



(e) Sample page of Parzival dataset



(f) Sample page of CESR dataset

Figure 2.1: Example of word spotting results (marked by rectangle box) in document images from 3 different datasets [1]: (a) GW; (b) Parzival; (c) CESR. The spotted query words are respectively shown in (a)(b)(c).

[1]For detailed description of the datasets, please see experimental evaluation section ***
For detailed description of Parzival dataset, please see: `http://www.iam.unibe.ch/fki/databases/iam-`

Depending on the usability criterion, we can classify word spotting techniques in different ways. A popular layman way to differentiate the word spotting techniques can be named as *query-by-example* and *query-by-string*. In the case of former category, a region of a document is defined by the user and the system should return all regions containing the similar text regions. In the later case, searching queries of arbitrary character combinations independently of their occurrence in the dataset can be possible, where some out of vocabulary words can also be searched.

2.2 Literature Review of Word Spotting Techniques

There have been numerous techniques proposed in the literature for word spotting. As mentioned before that these techniques could be broadly classified in two categories: *i) Query by string* and *ii) query by example*. The *query-by-string* approach requires a model for every character and these methods are often achieved by learning-based approaches, while *query-by-example* is often achieved by learning free, image matching based approaches. A well known drawback of learning based approaches is the requirement of a set of transcribed text line images for training and it may be costly to obtain for some particular historical datasets. Moreover, it should be done for all documents, considering variability of writing/font styles. Thus, if neither the language nor the alphabet of a historical document are known or even if writing style or font is different from the learned patterns, learning based word spotting approaches performs poorly. In this kind of situations, learning free word spotting approaches might be the only option available. So, a fair comparison is difficult to perform between learning based and learning free approaches as each of them have their own set of advantages and disadvantages. In the case of learning based approaches, ground truth (GT) is a mandatory requirement for training the system whereas in the case of learning free approaches, GT is not required and it could be more independent of data, language and scripts. As a counterpart, learning based approaches have most often high accuracy than learning free approaches and query-by-string based word spotting can be adapted in this category of techniques whereas learning free approaches is generally achieved by query-by-example based techniques.

2.2.1 Learning Free Word Spotting

In these categories of methods, a zone of interest (ZOI) is selected by the user, which correspond to the *query* image. Then the image dataset is compared against this ZOI patch and image regions similar to this ZOI are outputted by the system. These training free approaches can be categorized into two main categories: *i)* word segmentation based; *ii)* segmentation free approaches.

2.2.1.1 Segmentation Based Word Spotting

The pioneering work in word spotting was done by Manmatha et.al. [Manmatha, R. Chengfeng and Riseman, 1996]. Their approach relies on the segmentation of images into

historical-document-database/parzival-database

words (ZOI). In this approach, the word images are represented by a sequence of features. A sliding window is used for extracting them features. This kind of word image representations can be thought of as 2D signal, which can be matched using dynamic programming [Rath and Manmatha, 2003], [Rath and Manmatha, 2006], [Meshesha and Jawahar, 2008a], [Khurshid et al., 2012] based approaches. By maintaining almost the similar word spotting architecture, various other dissimilarity measures can be visible in literature e.g.: Scott and Longuet-Higgins distance measure [Manmatha, R. Chengfeng and Riseman, 1996], Hausdorff distance of connected components [Lu and Tan, 2002], etc. A pixel based comparison of the query and test images (word images extracted) can also be performed and a global dissimilarity measure between the two pixel sets is evaluated as an estimation of match between two images. A comparative experimental investigations of different image dissimilarity measure, such as shape context matching, XOR based comparison, sum of squared distances (SSD) correlation, Euclidean Distance Mapping, is given in [R. Manmatha, 2003]. The computational complexity and the discrimination between valid and invalid words are the shortcomings of these techniques, mentioned in [R. Manmatha, 2003]. Another kind of dissimilarity measure is proposed in [Rothfeder et al., 2003]. Here the sum of Euclidean distances of corresponding key points (corner features) obtained from query and target images are used as the dissimilarity measure between two word images. A similar architecture of word spotting technique is proposed in [Zhang et al., 2003] [Srihari et al., 2006]. The similarity between target and query image (GSC features, explained later) is calculated by using fast bit based operations for obtaining fast results. In [Yao et al., 2015], the author introduced two directional DTW and demonstrates that the use of two-directional DTW matching method for handwritten word spotting performs better than conventional DTW based word spotting techniques. A sequence of HOG features along rows and columns are calculated by extracting HOG descriptors from each cell of the normalized images and then these features are used for matching by two directional DTW. Since the quadratic time complexity of DTW is highly expensive for large scale word image matching, authors in [Nagendar and Jawahar, 2015], proposed a technique to approximate DTW distance as a sum of multiple weighted Eulidean distances for fast retrieval of word images. By learning a small set of global principal alignments from the training data and avoiding the computation of alignments for query images, this technique achieve 40 times speedup over classical DTW based approaches and shows similar results as classical DTW (bit lower).

As discussed in the above paragraph, many distance matching techniques have been explored in the literature for word spotting. Among all these mentioned distance matching approaches, classical DTW performs better than others. Although there are several other improved versions of classical DTW, which have shown better performance in other domains, but never been explored for the problem of word spotting. In Chapter 3, we will explore many of such improved DTW techniques for word spotting.

There have been many other researchers, who has worked for finding better set of features for representing word images, which can provide more meaningful and distinguishable information for word spotting. In [Zhang et al., 2003] [Srihari et al., 2006], gradient based binary features (GSC features) are calculated from query and target images and are used for word image matching in handwritten documents. The gradient, structural and concavity features captures multi-scale characteristics embedded in the image. The authors

of [Leydier et al., 2007], proposed an elastic matching technique for word spotting, by comparing pixel-wise, gradient based differential features of query and target images, in order to match only the informative parts of the words. In [Anurag Bhardwaj, Damien Jose, 2008], the authors have proposed a technique to calculate moments based, more complex and holistic features from the foreground pixels of the segmented word images. A set of feature vectors are obtained by using discrete cosine transform of the contour [Adamek et al., 2006] and these features are used for word image matching. Utilization of Gabor based features for word spotting was also investigated in [Cao and Govindaraju, 2007]. A sliding window of one pixel width is used to calculate several column based features from binarized images. These sort of features have been highly used by several researchers [Rath and Manmatha, 2003], [Rath and Manmatha, 2006], [Meshesha and Jawahar, 2008a], [Khurshid et al., 2012]. A sliding window based HOG features and block based HOG features were used by [Terasawa and Tanaka, 2009] and [Yao et al., 2015] respectively. Among all the mentioned features, HOG features shows better performance compared to others but column based features has also performed well. Although HOG based features slightly outperforms the column based features but the column based features have been widely used in literature due to their subsequent properties : *i)* easy and fast to compute; *ii)* low dimensionality; *iii)* well experimented and explored before. Moreover, the complexity in computation and high dimensionality are critical bottlenecks of HOG based word spotting systems. A methodology for using sequential data in conjunction with the holistic approach for word spotting in speech recognition domain is performed in [Keshet et al., 2009], where a sequence is transformed into a vector space and classified by using kernel machines. Although, the usage of different sets of features has shown some stirring improvements over earlier set of techniques, but the time required to calculate these new sets of features, along with the necessity of perfect word segmentation remains drawback of these methods. In the following section, we explain some of the techniques which uses different word spotting architecture than the ones based on feature extraction process.

A shape-based word image matching scheme, represented by local contour features is presented in [Giotis et al., 2015]. The proposed technique is accomplished in two steps. The query image is firstly aligned with the test image according to a similarity measure defined on their descriptors and then the aligned images are matched through a deformable non rigid point matching algorithm. One of the crucial drawback of this technique is it's high dependency on outcome of binarization process as an intermediate step. An attribute based approach that leads to a low dimensional, fixed length representation of the word images that is fast to compute and, especially, fast to compare is proposed in [Almazan et al., 2013]. A calibration technique is also proposed to correct the attributes scores based on canonical correlation analysis that greatly improves the results.

A shape coding based technique for word spotting is described in [Lu et al., 2008] [Tarafdar et al., 2010]. In this category of technique, each word image is denoted by a word shape code. By utilizing topological and morphological information, interest points are selected and an adapted version of shape context (SC) descriptor is employed on the handwritten texts. The final similarity measure is calculated by a weighted mixture of the SC cost, loop difference, stroke analysis and texture comparison with different weights. Based on the topological and morphological information of hand writing, a skeleton based graph matching technique is used in [Wang et al., 2014a], for performing word spotting in

handwritten historical documents. This structural representation is suitable for inherent deformations of handwriting. Dependency on perfectly segmented words and well quality binarization output is critical requirement of this technique. Another similar approach is proposed in [Riba and Llad, 2015]. Graphemes are extracted from shape convexities and are used for word spotting by associated them to graph nodes. Graph based word matching is defined by bipartite-graph matching algorithm.

A query by string paradigm for word spotting is proposed in [Aldavert et al., 2013]. In this work, a character n-grams based textual and a bag of visual words based representations are merged together to retrieve the query word images. In another work mentioned in [Roy et al., 2013], a bag of character n-gram based, recognition free image retrieval technique is proposed. This technique is able to search at sub word level and out of vocabulary words. This method inherently depends on generating a bag of *n-grams*, which is then used for off-line indexation and retrieval. This is a impediments of n-grams based word spotting approaches.

Perfect segmentation of words is a critical bottle neck of above mentioned systems. In the literature, there have been some attempt to spot words on segmented lines to avoid such problems. Depending on the documents quality, line segmentation could be comparatively easier than word segmentation. In such cases, a popular variant of DTW named as continuous dynamic programming (CDP) [Oka, 1998] is used for calculating the dissimilarity measures of image patches, hence the partial sequence matching property of CDP [Terasawa and Tanaka, 2009, Mondal et al., 2014] is utilized in such cases. In [Kolcz et al., 2000] also, a DTW based dissimilarity measurement techniques is utilized for word spotting on segmented lines.

Necessity of proper word segmentation (or line segmentation in some cases) and high computational complexities are the critical bottle-necks of most of the techniques in this category. Moreover these techniques are prone to usual degradation noise of historical document images and always require at-least one occurrence of query image in the dataset. To overcome the problem of segmentation, segmentation free approaches are proposed in the literature.

2.2.1.2 Segmentation Free Word Spotting

In this category the user defines a particular region of text image and the system is able to spot the similar regions in the completely unsegmented document images [Anurag Bhardwaj, Damien Jose, 2008]. A common approach for segmentation free word spotting is to consider the task as an image retrieval tasks for an input shape representing the query image as a part of full image [Moghaddam and Cheriet, 2009], [Leydier et al., 2005], [Gatos and Pratikakis, 2009], [Almazán et al., 2012]. HOG descriptors based sliding window protocol is used to locate the document regions that are most similar to the query. Another segmentation and classification free word spotting approach is presented in [Saykol et al., 2004] in which, a codebook of shapes is used to create a compressed version of each document. A word search is performed by using the stored shape codebook entries. In [Vasilopoulos and Kavallieratou, 2013], the queries are treated as compact shapes and by using image processing techniques, the query image is located in the document images. There have been also several notable work using Bag-of-Features paradigm [Fernandez-

Mota et al., 2014]. But recently, there is new trend of automatically learning descriptors from data. In [Sudholt et al., 2015], the authors has proposed a descriptor learning based pipeline for word spotting. Evaluation results demonstrates that word spotting results can effectively be improved by learning specialized local image descriptors. Currently this approach is been applied A heat kernel signature (HKS) based segmentation free word spotting technique is proposed in [Zhang and Tan, 2013]. By detecting SIFT based key points on the document pages and the query image, HKS descriptors are extracted from local patch centered at key points. Then a searching method is proposed to locate the local zones which contains enough matching key points corresponding to the query image. The performance of intrinsically variable hand writing, highly depends on correct detection of key points and corresponding features. The influence of key points selections and the associated features on the performance of word spotting process was also experimented in [Fernandez-Mota et al., 2014]. A bag of visual words (BOVW) [Dovgalecs et al., 2013] based approach is used to identify the zones of the image that share common characteristics with the query word. Then Longest Weighted Profile (LWP) based zone filtering technique is used to identify the location of query words in the document image. A patch based frame work, where local patches are described by a bag-of-visual-words model powered by SIFT descriptors is proposed in [Rusiñol et al., 2011], [Rusiñol et al., 2015]. By projecting the patch descriptors to a topic space with the latent semantic analysis technique and compressing the descriptors with the product quantization method, the approach is able to efficiently index the document information both in terms of memory and time. Another technique proposed in [Rothacker et al., 2013], where bag-of-visual-words based features [Rusiñol et al., 2015] are used for modeling a HMMs, for performing segmentation free word spotting. The discrete nature of this model enables to estimate a query model with only a single example of the query provided by the user. This makes the method very flexible with respect to the availability of training data. Segmentation free approaches are able to overcome the curse of segmentation problems but they have a comparative low accuracy (in comparison with segmentation based and learning based approaches) and high computational burdens considering full document image regions as an apparent candidate for matching.

2.2.2 Learning Based Word Spotting

The word spotting techniques in this category can be broadly classified into two sections: *i)* Hidden Markov Models based word spotting. *ii)* Bidirectional Long Short Term Memory based word spotting. There are only few other word spotting approaches, mentioned in [Kessentini and Paquet, 2015] [Almazan et al., 2013]. In the following section, we would focus mainly on these two main category of learning based word spotting techniques.

2.2.2.1 Hidden Markov Models based word spotting

The very first learning based approach for word spotting on poorly printed documents, was performed by Kuo et. al. [Agazzi and Kuo, 1994]. In their method, an initial layout analysis step is devoted to perform word segmentation from printed and handwritten document images. To represent the actual single word and all other segmented extraneous

words respectively, two statistical model named as Pseudo 2-D Hidden Markov Models (HMM) were used. Dynamic programming is then used for matching an unknown single input word with the help of these two 2-D HMM based models and finally maximum likelihood measure is calculated for correctly spotting the query word. The requirement for individual word images, segmented from the text region of the scanned document image is a primary bottleneck in the above mentioned word segmentation based word spotting approach.

To avoid this problem, line segmentation based word spotting approaches were introduced. By modeling imperfect word segmentation as probabilities and integrating the word segmentation probabilities into the word spotting algorithm, the method described in [Cao et al., 2009] proposes a generalized framework for word spotting. The word recognition scores are also converted into probabilities that are compatible with the probabilistic word spotting model. Word spotting based on HMM have been very popular in recent days [Lavrenko et al., 2004], [Chan et al., 2006], [Fischer et al., 2010a]. But an important drawback of these approaches is the large computational cost of the word specific HMM Viterbi decoding process, needed to obtain the confidence scores of each words to be spotted. The research work mentioned in [Toselli and Vidal, 2013], proposes a technique to calculate such confidence scores, directly from character lattices produced during a single Viterbi decoding process using only *"filler"* model. No explicit word-specific decoding is employed. Anyway, Viterbi decoding process remains computationally expensive and the learning needs ample amount of training data. The other notable work on HMM based word spotting was proposed in [Rodriguez and Perronnin, 2008]. In this work, the authors use unsupervised adaptation of whole word HMM to a specific writer. The usage of Fisher kernel of HMM for estimating a good confidence measure was explored by Perronnin and Rodriguez-Serron [Perronnin and Rodriguez-Serrano, 2009]. Given an word image and a keyword generative model, a vector can be generated which can describe how the parameters of keyword model should be modified to best fit the word image. The authors claim that their proposed system is 15 times faster than the baseline approach. A generalized HMM model for word spotting was proposed by Edwards et.al. in [Edwards, J. Teh, Y. W. Forsyth, D. Bock, R. Maire, M. Vesom, 2004]. In this research work, the author employs more than one emission in each hidden state and the results are obtained by using unigram, bigram, trigram models. In [Fischer et al., 2013], the learning based, lexicon free method with character n-gram language models (by using character HMM), has shown a high performance for word spotting. A document indexing and word spotting technique is performed by using semi markov conditional random fields (semi-CRF). This model can provide a framework for fusing the information of different contexts. For fast retrieval and to save storage space, the lattice is first pruned by forward and backward pruning approach. On the reduced lattice, the character similarity scores based on the semi-CRF model is estimated. A promising method for word spotting was also proposed in [Howe, 2013]. By treating the queries as compact shapes, this technique can infer a generative word appearance model from a single instance of query word. Later, by using this model, the system can retrieve similar words from arbitrary documents. The advantage of this system is that it requires minimal level of initial training.

Holistic word features in conjunction with two probabilistic, statistical, annotation model is used for retrieval of query images in large collections of handwritten manuscripts

[Rath et al., 2004]. Both of the model uses a set of transcribed page images to learn a joint probability distribution between features computed from word images and their transcriptions. In another similar probabilistic model based technique [Kessentini et al., 2013], the features extracted from segmented lines are used for training new filler based HMM model, which allows to speedup the decoding process.

2.2.2.2 Bidirectional Long Short Term Memory based word spotting

Along with the development of HMM based word spotting techniques, neural network based word spotting techniques has been attracting notable interest of the research community. Especially, Bidirectional Long Short Term Memory (BLSTM) based neural network [Fernández et al., 2007] [Wollmer et al., 2009] has been successfully used for word spotting. This learning based technique has a high similarity with the method mentioned in [Frinken et al., 2012], but the former methods primarily deals with word spotting in speech. Moreover, the designed architecture of the neural network symbolizes one node at the output layer as one word and it get triggered when the word occurs in the input data. Therefore the number of words to be spotted are limited as the particular word has to be known beforehand and the particular word must occur in training. But the later method, mentioned in [Frinken et al., 2012], employs a template free word spotting technique. The word spotting is done by using a modification of the Connectionist Temporal Classification (CTC) Token Passing algorithm in conjunction with a recurrent neural network.

Although learning based techniques shows comparatively better accuracy than learning free approaches, but the necessity of training for every different types of script, glyphs remains a big bottle-neck of learning based techniques. Unfortunately these HMM and neural network based techniques requires sufficient amounts of training data and considerable training time to perform, which could be difficult to obtain in some conditions.

2.2.3 Summary of Word Spotting Techniques

In the following Table.2.1, we tried to categorize the various approaches for word spotting. *Learning* is denoted as column wise and levels of *Segmentation* is considered row wise (*"No Segmentation", "Line based segmentation", "Word based segmentation", "Character based segmentation"*). As mentioned earlier, our contributions belong to the category of learning free word spotting based on segmented lines or improperly segmented words. Since our contributions are extensions of sequence matching (DTW, MVM etc.) techniques, their use for word spotting is based on a classical framework, similar to the one used previously [Rath and Manmatha, 2003], [Rath and Manmatha, 2006], [Meshesha and Jawahar, 2008a].

2.3 General Architecture for Sequence Matching Based Word Spotting

In the following section, we introduce our word spotting architecture. This is a general and commonly used word spotting architecture, which has been used by several other

Table 2.1: The grouping of state of the art techniques, available in literature.

	Without Learning	**With Learning**
Segmentation Free	[Anurag Bhardwaj, Damien Jose, 2008] [Moghaddam and Cheriet, 2009], [Leydier et al., 2005], [Gatos and Pratikakis, 2009], [Almazán et al., 2012] [Saykol et al., 2004] [Vasilopoulos and Kavallieratou, 2013] [Zhang and Tan, 2013] [Dovgalecs et al., 2013] [Rusiñol et al., 2011], [Rusiñol et al., 2015] [Rusiñol et al., 2015]	[Rothacker et al., 2013]
Line Segmentation Based	[Terasawa and Tanaka, 2009, Mondal et al.,]	[Agazzi and Kuo, 1994] [Cao et al., 2009] [Lavrenko et al., 2004] [Chan et al., 2006], [Fischer et al., 2010a] [Toselli and Vidal, 2013] [Perronnin and Rodriguez-Serrano, 2009] [Agazzi and Kuo, 1994] [Edwards, J. Teh, Y. W. Forsyth, D. Bock, R. Maire, M. Vesom, 2004] [Rodriguez and Perronnin, 2008] [Fischer et al., 2013] [Howe, 2013] [Rath et al., 2004] [Kessentini et al., 2013] [Fernández et al., 2007] [Wollmer et al., 2009] [Frinken et al., 2012]
Word Segmentation Based	[Manmatha, R. Chengfeng and Riseman, 1996] [Rath and Manmatha, 2003], [Rath and Manmatha, 2006], [Meshesha and Jawahar, 2008a], [Khurshid et al., 2012] [Lu and Tan, 2002] [R. Manmatha, 2003] [Rothfeder et al., 2003] [Zhang et al., 2003] [Srihari et al., 2006] [Leydier et al., 2007] [Anurag Bhardwaj, Damien Jose, 2008] [Adamek et al., 2006] [Cao and Govindaraju, 2007] [Keshet et al., 2009] [Almazan et al., 2013] [Lu et al., 2008] [Tarafdar et al., 2010] [Wang et al., 2014a] [Aldavert et al., 2013] [Kolcz et al., 2000]	
Character Segmentation Based	[Roy et al., 2013]	

researchers [Rath and Manmatha, 2003], [Rath and Manmatha, 2006], [Meshesha and Jawahar, 2008a], [Khurshid et al., 2012] [R. Manmatha, 2003] [Terasawa and Tanaka, 2009], [Mondal et al., 2014]. Most of the sequence matching techniques, introduced later, are experimented based on this following word spotting architecture. For evaluating the performance of each individual sequence matching technique, we only change the matching techniques in the system diagram/architecture shown in following Fig. 2.2. In this section, our complete word spotting framework is briefly explained in stepwise manner. It is similar

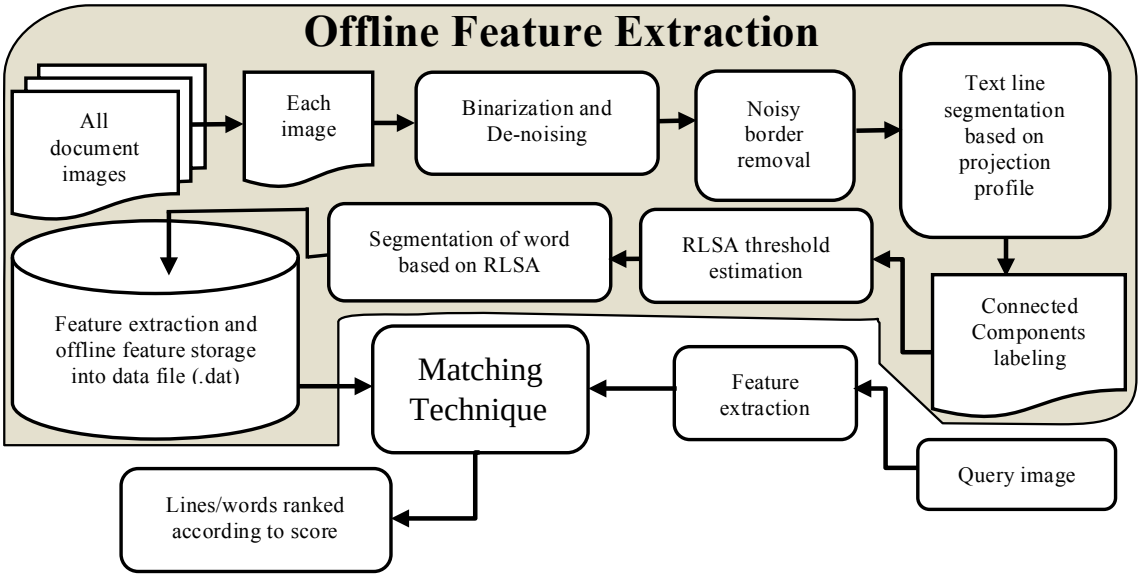


Figure 2.2: The block diagram of word spotting system

to the one that has been used by several researchers [Rath and Manmatha, 2003, Rath and Manmatha, 2006, Khurshid et al., 2012, Meshesha and Jawahar, 2008a].

2.3.1 Image Processing and Segmentation

Firstly, we need to preprocess each of the scanned document pages. As the performance of popular binarization techniques (e.g Otsu's [Nobuyuki, 1979] technique) is not good enough for old historical manuscripts, we decided to use the adaptive binarization technique proposed in [Gatos et al., 2006]. Due to improper scanning, document images might be framed with unwanted text areas of the neighboring pages. So, after document binarization, we apply the technique, described in [Stamatopoulos et al., 2010], for unwanted text region removal and for obtaining proper text boundary.

After preprocessing and extraction of textual regions, the regions have to be segmented either into lines or pieces of lines, up to words or part of words. Here we named this textual elements as ***"pseudo word"***. Although this process of word segmentation technique is a basic one and it gives many segmentation errors (mainly under segmentation) but our proposed matching technique (discussed later in Section ***) is designed for handling these kinds of problems. Depending on the quality and level of difficulties to segment documents, either line or word extraction can be used (even a mixed of both). It is noteworthy to mention that for our experimentations, we used line segmentation for some specific datasets and word segmentation information for other datasets. One advantage in the case of line segmentation is that, it can be possible to spot hyphenated words spanned into two lines, simply by concatenating them. The system explained here, is mainly focused to describe our adapted ***pseudo word*** segmentation technique for printed historical documents.

Word segmentation could be obtained by using basic Run Length Smoothing Algorithm

(RLSA) based segmentation techniques. This one can provide good results for high quality printed documents, but the results could be poor when inter-word spaces are variable which is often the case in handwritten documents as well as in historical printed documents. Horizontal Run Length Smoothing Algorithm (H-RLSA) with adaptive threshold [Nikolaou et al., 2010]. This threshold actually defines the average inter character gap in a word. This threshold inherently depends on the considered dataset. To obtain it automatically, a preliminary text line segmentation is performed on the document pages. Proper segmentation of all the text lines is not at all required in this case. From each page, we need to have only some prominent segmented text lines, which can help us to understand the inter characters gap in words and the inter words gap in a text line. So, basic text line segmentation algorithm (based on projection profile [Nikolaou et al., 2010]) is used here. As the line segmentation technique is not robust enough, only coherent segmented lines (based on the height of segmented text lines) are taken into account. After finding the most prominent text lines, the average inter word's character gap is obtained by finding gap histogram between connected components [Haralick and Shapiro, 1992] (CCL) from each line. Hence the inter character gaps present in maximum number of component pairs is calculated and average distances between the characters present in a word are obtained. After that (HRLSA) is performed on the image by using this average character gaps as the threshold. As a result, all characters belonging to a word are merged so that the words boundaries can be recognized for segmentation.

2.3.2 Feature Extraction

After segmenting the document image pages into pseudo words or lines, the next task is to extract the useful features from these ones. Features are extracted from gray scale and also binary normalized images. All the pseudo word images are normalized by equalizing height of all the images. In the following section, we describe two different category of features, namely column based features and histogram of gradient based features that are used, throughout our experimental process. Depending on the suitability and robustness of the features, these ones are applied for experimenting each individual datasets.

2.3.2.1 Column Based Feature Extraction

A set of statistical column based features, which have been used previously for handwriting recognition is broadly described in [Marti and Bunke, 2001]. A subset of these features have been used by several researchers for word spotting [Rath and Manmatha, 2003], [Rath and Manmatha, 2006], [Khurshid et al., 2012], [Frinken et al., 2012], [Rodríguez-Serrano and Perronnin, 2009], [Fischer et al., 2012]. Although these features can be outperformed in terms of accuracy by more complex features, e.g. gradient based features [Rodríguez-Serrano and Perronnin, 2009], graph similarity features [Fischer et al., 2010b] etc., due to their less computational cost they remains quite interesting. Here, we have chosen 8 features, $F_1, F_2, \ldots, F_8$ to describe each pixels column. Thus, for an image of N pixel's width, a sequence (size N) of 8 dimensional feature vectors are obtained by moving from the left to right direction, over the segmented pseudo word image. The description of the features is given below in Table 2.2. Among these features, the features $F1 - F6$ have

been used several times in literature [Rath and Manmatha, 2003], [Rath and Manmatha, 2006], [Khurshid et al., 2012], [Frinken et al., 2012], [Rodríguez-Serrano and Perronnin, 2009], [Fischer et al., 2012], but the features F7 and F8 are proposed in this work.

The feature F7, corresponds to the center of gravity of foreground pixels inside a column. In the corresponding equation Table.2.2, w_b denotes the binarized version of the word image and Y represents row's coordinates. It is note worthy to mention that, the features for the columns having no foreground pixels are calculated by nearest neighbor interpolation, with the help of the neighboring columns having foreground pixels. The feature F8, is calculated by using these center of gravity location, obtained from F7: the number of transition, from foreground to background (1 to 0) or from background to foreground (0 to 1) at these calculated centroid location of each pixels. All these features, are computed and stored off-line for faster computation.

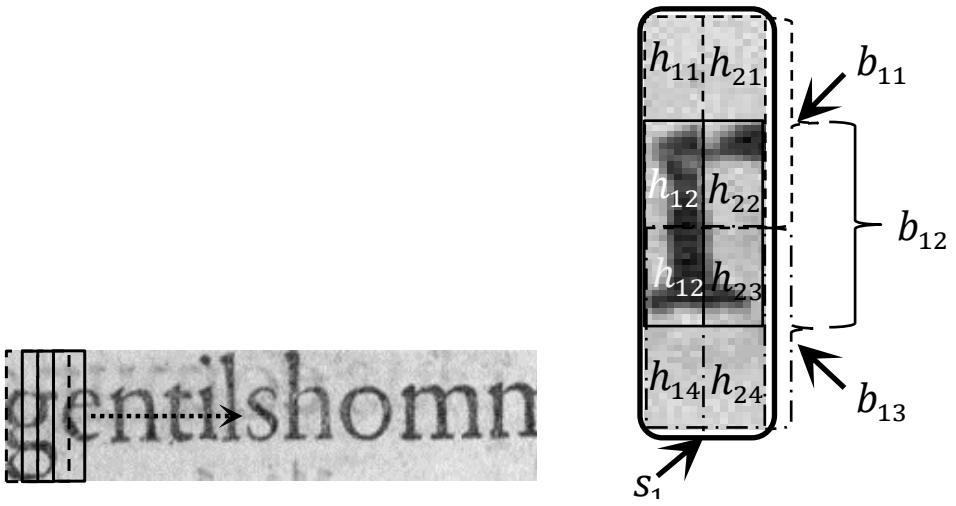
Table 2.2: Extracted features from the word images, considering an image with N columns and M rows

Sr. No	**Feature set description**
F1.	Projection Profile of foreground pixels in each column
F2.	Background-to-ink transition in pixel column
F3.	Upper Profile of foreground pixels in each column
F4.	Lower Profile of foreground pixels in each column
F5.	Distance between upper and lower Profile
F6.	Number of foreground pixels in pixel column
F7.	Center of gravity (C.G.) of the column obtained from the foreground pixels ($1 \leq n \leq N$) $F7(n) = \begin{cases} [\frac{1}{\rho}\sum_{m=1}^{M} m \quad if\ w_b(m,n)=1]; & \rho \neq 0; \rho = No.\ of\ foreground\ pixels\ at\ n^{th} column; \\ t \quad \text{Obtained by interpolation;} & \rho = 0 \end{cases}$
F8.	Transition at C.G. obtained from F7 $F8(n) = \begin{cases} 1 & w_b(F7(n),n)=0; \text{ and } w_b(F7(n-1),n)=1 \text{ or} \\ & w_b(F7(n),n)=1; w_b(F7(n-1),n)=0 \\ t & \text{Obtained by interpolation} \end{cases}$

2.3.2.2 Slit Style HOG Based Feature Extraction

For the task of word spotting, a slit style HOG (SSHOG) [Dalal and Triggs, 2005] based feature extraction technique can be used. This slit style HOG (SSHOG) is a modified version of HOG, to make it suitable for word spotting applications.

A fixed sized slit window is slided over the image in horizontal direction for extracting features from each slit. In the following Fig. 2.3a, the demonstration of slit style feature extraction process is described. HOG computes a histogram of gradient orientations in a certain local region. HOG features are computed in a rigid rectangular window without scale/orientation normalization and it calculates normalized histograms in overlapping local blocks. For calculating the HOG descriptors, we need to divide the image into smaller rectangular regions (called *cells*) along horizontal and vertical direction. Let's say the whole image is divided into $H \times W$ cells. Then the number of bins into which the weighted votes of the gradient vectors should be accumulated is decided. Let's consider here, ξ denotes the number of orientation bins. Thus, from $H \times W$ cells, we could obtain a histogram



(a) (b)

Figure 2.3: (a) Feature extraction using slit style sliding window. (b) Block normalization technique for SSHOG, where S_1 is a slit and b_{11}, b_{12} & b_{13} are blocks that overlaps.

with $HW\xi$ bins. After obtaining the histogram bins, a block normalization is performed. A block is defined as a group of $h \times w$ cells. So, from each block, we can generate $hw\xi$ dimensional vectors by concatenating the histogram components of each cell. A block can slide in vertical and horizontal directions. If we assume that a block slides $h-1$ cells in vertical and $w-1$ cells in horizontal directions, then there are total $(H-h+1 \times W-w+1)$ unique block exists. The HOG descriptor of considered image portion, is a concatenation of the normalized block descriptors, obtained from this image region. Consequently, HOG descriptor has $(H-h+1)(W-w+1)hw\xi)$ dimensionality. It can be understood that, this is a redundant expression in a sense that $HW\xi$ components in the original histogram composes a vector with $(H-h+1)(W-w+1)hw\xi)$ dimensions, but this is a inherent characteristics of HOG descriptor.

In order to make it suitable for word spotting application, some modification is made to the HOG descriptor. Our window image is a narrow rectangle, where each sliding block, which slides over each slit is of same width as the width of the block. The horizontal overlapping of the original HOG could be well realized by the sliding window and sequential representation of vectors. The representation and relationship between slit, blocks and cells are shown in following Fig. 2.3b. The figure shows, 3 blocks as b_{11}, b_{12}, b_{13} with each block composed of 4 cells (2×2). Hence, the dimensionality of slit style HOG feature becomes $3 \times 4 \times \pi$. For our experiment, we used signed gradient instead of unsigned gradient for the orientation binning as signed gradient shows better results as the portions of characters in the image is brighter than background regions.

2.4 Fast Word Image Retrieval Technique Based on Kernelized Sensitive Hashing (KLSH)

In the previous sections (refer to Section 2.2), we have seen several word spotting or word image matching systems. But the domain of fast word image retrieval has been less explored in literature. The generalized approach for image matching is computationally expensive, especially if matching is to be done over a very large amount of document images. In this research work, we performed a preliminary investigation of hashing with word images. The following mentioned work is an elementary level of contribution, where we analyze the performance of one well known hashing technique (Kernelized Sensitive Hashing (KLSH) [Kulis and Grauman, 2009]) by using a set of holistic features.

In this work an efficient approach to index and retrieve word images for large document image database is stated. For the word indexing and retrieval, the most basic and essential task in image search is the *"nearest neighbor"* technique: given a query image, the task is to accurately search examples, which are most similar to it. In approximate similarity search technique (like LSH [Indyk and Motwani, 1998] [Charikar, 2002]) for high dimensional input data, accuracy is sacrificed to some extent for allowing fast retrieval. The basic idea of LSH is to calculate a randomized hash function that guarantees a high probability of collision for similar examples. There are certain other works to show how to form low dimensional binary embedding for capturing more expensive distance function [Torralba et al., 2008] [Weiss et al., 2009]. The existing techniques generally assume that the data to be hashed, comes from a multi-dimensional vector space and also requires underlying embedding of the data to be explicitly known and computable. Like LSH, that relies on random projection of input vectors.

Kernelized locality sensitive hashing [Kulis and Grauman, 2009] is used to quickly retrieve word images, based on the extracted word image features. The high dimensional features are extracted and embedded into a low dimensional hamming space, so that items can be searched efficiently and quickly. Existing methods are not useful for high dimensional kernelized data, when the underlying kernel for feature embedding is unknown or very expensive to know. The main technical contribution of KLSH is to generalize LSH for accommodating arbitrary kernel functions for unknown or computationally expensive feature space embedding, thereby offering sub-linear time similarity search. The method is not dependent on data distribution or input, so it has wide spread applicability for many successful image-based kernels, having unknown or incomputable feature space embedding. The problem demonstrated in KLSH is as follows: given a kernel function $\mathfrak{K}(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T, \mathbf{x}_j)^T$ and database of n objects, how can we quickly find the most similar items to a query object q in terms of kernel function, that is $\arg\max_i \mathfrak{K}(\mathbf{q}_i, \mathbf{x}_i); i = 1, 2, ...n$. Like standard LSH, hash functions of KLSH involve computing random projections; however, unlike standard LSH, these random projections are constructed using only the kernel function and a sparse set of examples from the database itself. In this section, we investigate how KLSH can be used to address some of the issues involved in effective and efficient retrieval of word images. We demonstrate that efficient retrieval could be done on a collection of George Washington (GW) database (explained later in Section 2.5.1) using the proposed technique [Mondal et al., 2009].

2.4.1 Feature Extraction for Hashing

Following features are extracted as the representation of word images. Please note that these features are not optimized ones as we have not experimented or explored other kinds of features. Based on prior knowledge in the domain of document image processing and motivation from state-of-the-art word image matching techniques [Cao and Govindaraju, 2007] [Manthalkar et al., 2003], [Marti and Bunke, 2001], [Rath and Manmatha, 2006], [Frinken et al., 2012], we decided to use these features for having a preliminary experience of hashing for word images.

2.4.1.1 Gabor Features

In order to get inherent information of the image, Gabor features are extracted from height normalized word images. Gabor kernel [Wang et al., 2002] is suitable for rotation and scale invariant feature extraction.

$$g(x, y, F, \sigma_x, \sigma_y) = \frac{1}{2\pi\sigma_x\sigma_y} e^{[-\frac{1}{2}(\frac{x^2}{\sigma_x^2}+\frac{y^2}{\sigma_y^2})+2\pi jFx]} \quad (2.1a)$$

$$\mathscr{G}(u, v) = e^{-\frac{1}{2}[\frac{(u-F)^2}{\sigma_u^2}+\frac{v^2}{\sigma_v^2}]} \quad (2.1b)$$

Where, $\sigma_u = \frac{1}{2\pi\sigma_x}$ and $\sigma_v = \frac{1}{2\pi\sigma_y}$ and F specifies the central frequency of interest. For localized frequency analysis, it is desirable to have a Gaussian envelope, whose width adjusts with the frequency of the complex sinusoids. We have considered a class of self-similar functions, referred as Gabor wavelets [Cao and Govindaraju, 2007] [Manthalkar et al., 2003]. Gabor wavelet derived from Eq^n 2.1a is defined as follows:

$$g_{(m,n)}(x, y) = a^{-m} g(x', y') \quad (2.2a)$$

$$x' = a^{-m}(x.cos\theta + y.sin\theta) \quad (2.2b)$$

$$y' = a^{-m}(-x.sin\theta + y.cos\theta) \quad (2.2c)$$

Where, $\theta = \frac{n\pi}{K}$ and K is the total number of orientations. The scale factor a^{-m} in Eq^n 2.2a is meant to ensure that the energy is independent on m.

$$E_{(m,n)} = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} g_{(m,n)}(|x - y|)^2 dxdy \quad (2.3)$$

This ensures that all filters in the set have the same energy. The non-orthogonality of Gabor wavelets implies that there is redundant information in the filtered images, and the following strategy is used to reduce this redundancy. Let U_h and U_l denote the lower and upper centre frequencies of interest. Also let K be the number of orientations and S be the number of scales in the multi-resolution decomposition. Then the design strategy is to ensure that the half peak magnitude cross-sections of the filter responses in the frequency spectrum touch each other. This results in the following formulas for computing the filter parameters a, σ_x and σ_y (and thus σ_u and σ_v).

$$a = (\frac{U_h}{U_l})^{\frac{1}{S-1}}; \sigma_u = \frac{(a-1)U_l}{(a-1)\sqrt{2\log 2}}$$
$$\sigma_v = tan(\frac{\pi}{2K})[U_l - \frac{2\log 2\sigma_u^2}{u_l}][2\log 2 - \frac{(2\log 2)^2\sigma_u^2}{U_l^2}]^2 \quad (2.4)$$

In order to make the Gabor filter sensitive to the strokes of the characters, the central frequency of interest should be set to $\frac{1}{2W}$, where W is the stroke width. This is because the stroke width is the half-period of the signal of interest, so the period is $2W$, and the frequency is $\frac{1}{2W}$. In GW dataset, stroke width (manually calculated) varies from 12 to 15 pixels, which corresponds to frequencies of 0.0625 and 0.0333, respectively. The upper and lower frequencies of interest are $U_h = 0.0625$ and $U_l = 0.0333$, so the range of stroke width can be completely covered. In this experiment, 5 frequencies $(0.0625, 0.0125, 0.25, 0.5, 1.0)$ and 7 orientations $0°, 30°, 60°, 90°, 120°, 150°, 180°$ are considered and both odd & even Gabor wavelets are considered. We have taken the mean and standard deviation of the image generated by applying Gabor wavelets of different orientation, scale and frequency. Hence a total of 280 features ($2\times5\times7\times2\times2 = 280$) are extracted. Experiments conducted indicate a better performance using 2-scale Gabor wavelet over single scale Gabor filter.

Table 2.3: Extracted column based features from the word images.

Sr. No	Feature set description
F9.	Standard Deviation of the intensities of pixels in a column
F10.	Skewness of the intensities of pixels in a column
F11.	Kurtosis of the intensities of pixels in a column
F12.	0^{th} order moment of the intensities of pixels in a column
F13.	1^{st} order moment of the intensities of pixels in a column
F14.	Central moments of the intensities of pixels in a column

2.4.1.2 Image Column Based Features for Hashing

Along with the column based features mentioned in Table 2.2, we use some more column based features (see Table 2.3). After calculating total 14 column based features from each column, the mean and standard deviation of each feature are calculated for the entire columns present in the word image. So, we will get $28(14 \times 2)$ more features and in total $310(280+28)$ features are obtained for each image. In the next section, it is described that how these features are used for generating hash table for fast retrieval of word images.

2.4.2 Kernelized Locality Sensitive Hashing (KLSH)

In this section, we give a brief overview of KLSH. Let $\mathbf{X} \in \mathbb{R}^{d\times n}$ be the collection data points to be searched and the query is represented by $q \in \mathbb{R}^d$. To efficiently search k nearest neighbors, LSH projects each data points into a low dimensional binary space, referred as *hash key*. This hash keys are constructed by applying b binary hash functions $h_1, .., h_b$ to the data points in $\mathbf{X}$. KLSH generalizes LSH by introducing kernel function $\kappa(\mathbf{x}_i, \mathbf{x}_j)$ for

mapping a data point x_i to a functional space through a nonlinear feature mapping $\phi(\mathbf{x}_i)$ that satisfy the condition $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \phi^T(\mathbf{x}_i)\phi(\mathbf{x}_j)$. To build a hash function, KLSH first randomly selects a subset of p data points from $\mathbf{X}$, denoted as $S = \{\mathbf{x}_1^s, ..., \mathbf{x}_p^s\}$ and forms a kernel matrix K over the sampled data points; it then generates b random vector $e_S^1, ..., e_S^b$ and computes a hashing function for each random vector $e_S^k (1 \leq k \leq b)$ as

$$h_k(\phi(\mathbf{x})) = sign(\sum_{j=1}^{p} w_j^k \kappa(\mathbf{x}, \mathbf{x}_j^s)) \tag{2.5}$$

where $\mathbf{w}^k = (w_1^k, ..., w_p^k)^T$ is given by $w^k = K^{-\frac{1}{2}} e_S^k$. The key steps of KLSH is outlined in Algorithm 1, where b is a critical parameter that determines the length of hash key to be constructed in KLSH.

Algorithm 1: KLSH ALGORITHM

Input: $\{\mathbf{x}_i | i = 1, ..., n\}, \mathbf{x}_i \in \mathbb{R}^d$(image database, containing multiple images)
Input: b (length of hash key)
Input: $\kappa(.,.)$(the kernel function)
Output: $\mathscr{H} = \{h_k | k = 1, ..., b\}$ (a set of b hash functions)

randomly choose p data points $\{S = \mathbf{x}_j^s | j = 1, ..., p\}$;
$K = [\kappa(\mathbf{x}_i^s, \mathbf{x}_j^s)]_{p \times p}$
for *k = 1,...,b* **do**
    form $\mathbf{e}_S^k$: select t indices at random from$[1, ..., p]$
    form $\mathbf{w}^k = K^{-1/2}\mathbf{e}_S^k$
    generate the hash function by: $h_k(\phi(\mathbf{x})) = sign(\sum_{j=1}^{p} w_j^k \kappa(\mathbf{x}, \mathbf{x_j^s}))$
return *bool*

2.4.3 Results

By considering different threshold values of nearest neighbors, the capability of indexing and retrieval of the dataset is demonstrated in Fig. 2.5a. In Fig. 2.4, we present some qualitative query images and their corresponding retrieved images. The method is tested with randomly chosen 30 images (which were not used for training) from database and matching result is obtained by averaging the accuracy of these selected 30 query images. This process of testing the robustness of the system has been repeated several times and results are obtained as given in Fig.2.5. For our experimentation, b = 300 and t = 30 (these values are experimentally obtained) are used.

Concerning the quantitative evaluation of GW dataset, the accuracy of the system increases exponentially with the increase of NN threshold, which is quite obvious but the notable point here is that almost 50% retrieval accuracy can be obtained by considering NN equal to only 20% of the total number of elements without doing any sort of pre-

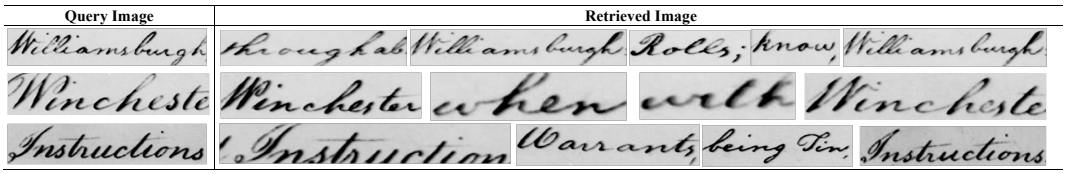
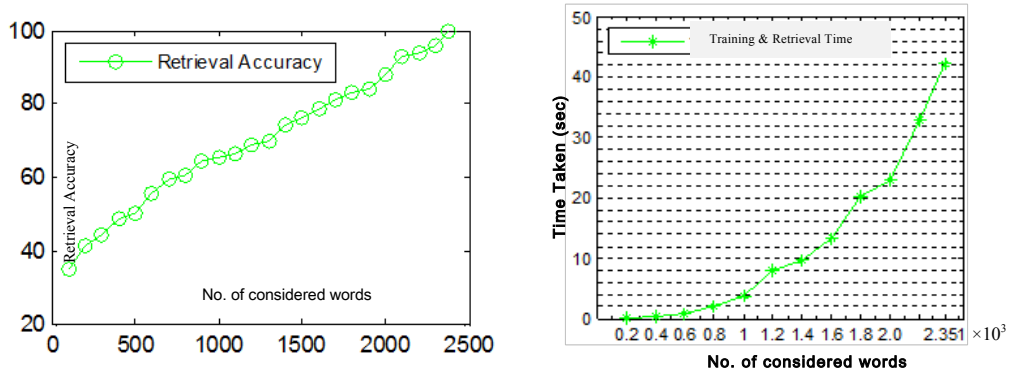


Figure 2.4: Some of the query image and their similar images received.

processing or pruning of word images. As shown in Fig. 2.5a, the accuracy of the system is far from being high but the advantage of this method is that the searching/retrieval time is comparatively much less than any other retrieval techniques mentioned in literature. So this can be taken as a trade-off between the accuracy and speed. The experiments were conducted on data sets of increasing size (by 200 words) at each iteration, and the time taken for training and retrieval of one word in shown in Fig 2.5b. The process of searching 30 query words against full dataset (i.e. 2351 words) takes 0.1807 second and searching of one word takes 0.0059 sec. (using a machine with Intel i5 processor and 4 GB RAM). Our prototype is implemented using Matlab, the speed can be increased using other low level language and optimized implementation. Application of locality sensitive



(a) (b)

Figure 2.5: (a) Retrieval accuracy v/s nearest neighbor threshold. (b) Time taken to train with certain group of words and retrieve one word from the database.

hashing permits sub linear time approximation based similarity search. KLSH significantly widens the paradigm of LSH into more generic fashion, as it focuses on unknown kernelized visual data and does not require assumptions about the data distributions or input. The result could be improved by pre-processing and pruning some irrelevant word images based on their size and ascenders/descenders, but we didn't do it for exploring robustness of the system. Results can be further improved by applying more suitable and prominent features for representing the characters written in the word images, which innately can improve the accuracy of the system.

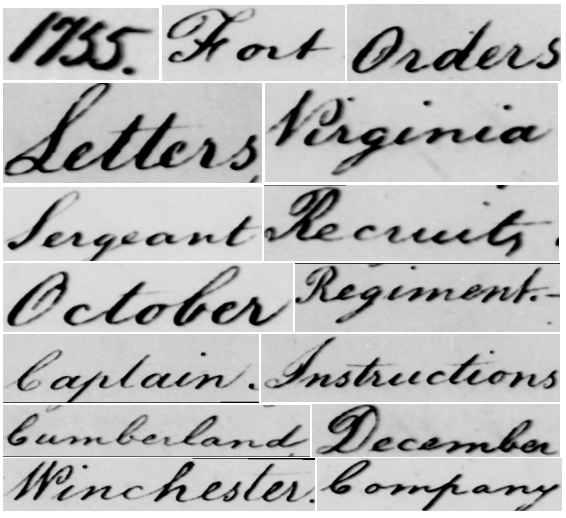
In the following section, we have explained various word spotting datasets, used throughout this thesis for various experiments.

2.5 Word Spotting Datasets

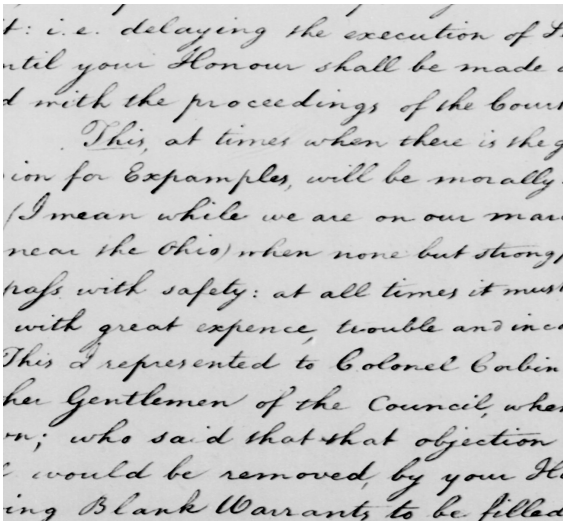
Several datasets can be used to evaluate word spotting algorithms, including the benchmark algorithms. In the following section, the datasets that we will use are described in detail. In this thesis we mainly explored word spotting through *"Query-by-example"* paradigm. In this kind of approach, the query image is selected by user or manually cropped by user and the system finds this query image inside full document page. Generally there are two main direction of *"Query-by-example"* based word spotting i) Segmentation based ii) Segmentation free.

2.5.1 George Washington Dataset

This dataset is the handwritten manuscript of *George Washington(GW)*, consisting of 20 pages of letters, orders, and instructions of *George Washington* from 1755 century. The quality of scanned pages varied from clean to difficult to read by human and the pages originates from large collection with a variety of images. For our experiment, we considered all the 20 pages. The text in the pages is a part of a larger corpus, written not only by *GeorgeWashington* but also some of his associates. So, obviously it inhibits some variation in writing style. We performed the experiments with GW dataset in two manners: i) experiment on properly segmented words from provided ground truth in the dataset; ii) experiments on segmented lines, by using SSHOG features, provided by Terasawa. et.al. [Terasawa and Tanaka, 2009]. In Fig. 2.6a, some query images are shown and in Fig. 2.6b a sample scanned page is shown from GW dataset.



(a) Some query images used for GW dataset



(b) Sample page from the GW dataset

Figure 2.6: Some query images and sample scanned page image from GW dataset

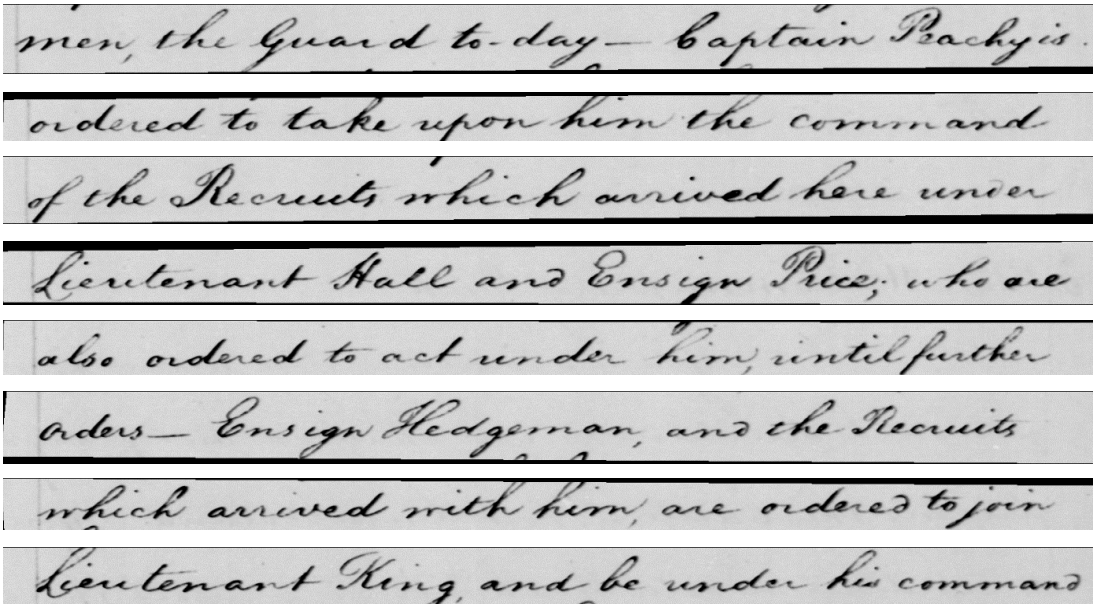
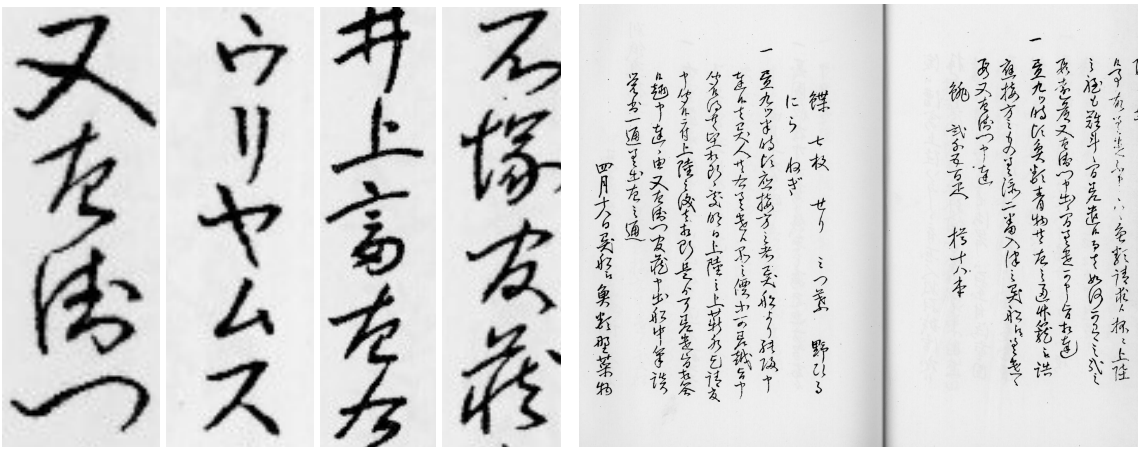


Figure 2.7: Some segmented text lines from GW dataset.

2.5.2 Japanese Dataset

The other experimental material is *Japanese* manuscript. It is consisting of scanned images of *"Akoku Raishiki (The diary of Matsumae Kageyu)"*. This dataset contains total 92 scanned images scanned in 72×72 ppi. There are total 1576 segmented lines available from the images. The Fig. 2.8 illustrates queries as well as a sample page. In this dataset, we used the GT provided by Terasawa. et.al. [Terasawa and Tanaka, 2009], which is based on segmented lines. The extracted SSHOG features from each of these lines are also provided.



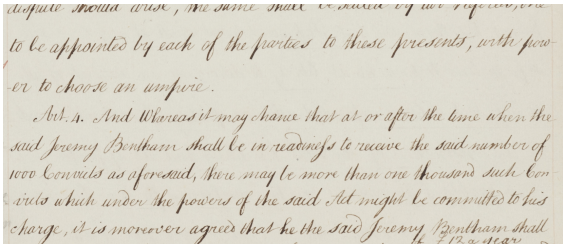
(a) Some query images used for Japanese dataset (b) Sample page from the Japanese dataset

Figure 2.8: Some query images and sample scanned page image from Japanese dataset

2.5.3 Bentham Dataset

This dataset [Gatos et al., 2014] consists of a series of documents from the Bentham collection. This dataset have been prepared in the tranScriptorium project[2]. The Bentham dataset mainly includes manuscripts written by Jeremy Bentham (1748-1832) himself over a period of sixty years, as well as fair copies written by Bentham's secretarial staff. The sample page and some of the query images are shown in Fig.2.9. The GT of this dataset is provided in word level.

(a) Query images used for Bentham dataset



(b) Sample page from the Bentham dataset.

Figure 2.9: Query images and sample scanned page image from Bentham dataset.

2.5.4 CESR Dataset

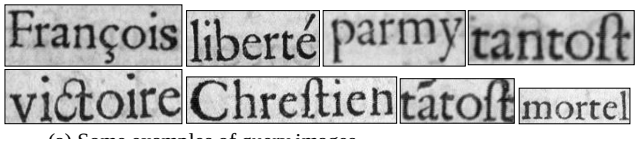
We have also applied our word spotting technique on a machine printed historical dataset, named as CESR[3] dataset. This datasets comes from the resources of the Centre d'Etude Suprieure de la Renaissance (CESR), through the BVH (Bibliothèques Virtuelles Humanistes[4]) Project. The CESR has a collection of precious historical books, dating from the middle of the XIV century to the beginning of the XVII century. The languages used in the books are Latin or French. The dataset was composed from the two volumes of *Essais de messire Michel Seigneur de Montaigne, Chevalier de l'order du Roy, & Gentilhomme ordinaire de Sa Chambre*. This first edition was published in Bordeaux in 1580 by S. Millanges [5]. All the pages of the book are scanned with a resolution of 312 × 312 dpi and saved in grey scale format with .jpg extension. The Vol.I of the book has 520 pages and the Vol.II of the book has 676 pages. To process this book, we used the preprocessing steps explained in Section 2.3. Consequently pseudo words (i.e. words, piece of words or piece of lines) were extracted with HRLSA and characterized with column based features. Some of such segmented pseudo words are shown in Fig. 2.10b.

[2]For the detail of this dataset, please see ICDAR-2015 *"Keywords spotting"* competition website : http://transcriptorium.eu/ icdar15kws/data.html

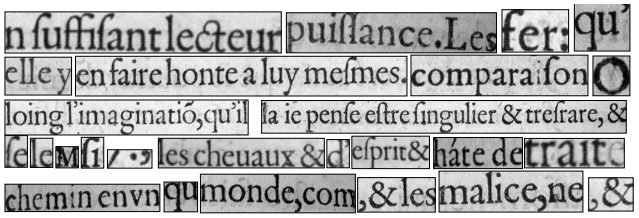
[3]http://cesr.univ-tours.fr/

[4]See Bibliothèques Virtuelles Humanistes Project: http://www.bvh.univtours.fr/presentation en.asp

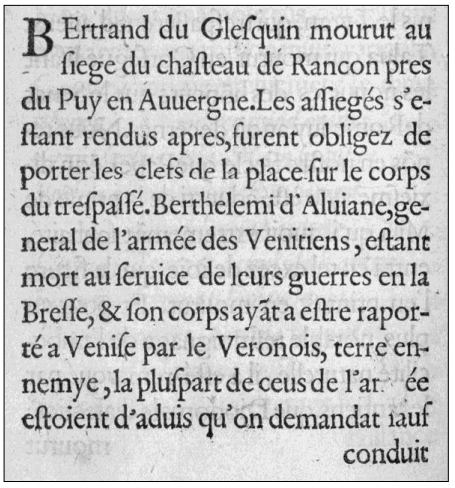
[5]Please see the links for the more details about the book : `https://www.lib.virginia.edu/rmds/collections/gordon/literary/montaigne/bibliography.html` and `http://search.lib.virginia.edu/catalog/u50318`



(a) Some query images used for CESR dataset.



(b) Some examples of segmented pseudo word images.


BErtrand du Gleſquin mourut au ſiege du chaſteau de Rancon pres du Puy en Auuergne. Les aſſiegés s'eſtant rendus apres, furent obligez de porter les clefs de la place ſur le corps du treſpaſſé. Berthelemi d'Aluiane, general de l'armée des Venitiens, eſtant mort au ſeruice de leurs guerres en la Breſſe, & ſon corps ayãt a eſtre raporté a Veniſe par le Veronois, terre ennemye, la pluſpart de ceus de l'ar ée eſtoient d'aduis qu'on demandat ſauf conduit

(c) Sample page from the CESR dataset

Figure 2.10: Query images, some properly and improperly segmented word images and sample scanned page image from CESR dataset.

2.6 Conclusion

We introduced this chapter by explaining a popular problematic domain called word spotting and we discussed the technique to represent word spotting or word image matching as a time series matching problem. After describing the domain of word spotting, we have categorically mentioned a broad overview of related literatures on word spotting problem followed by the description of our word spotting framework. The literature review reveals that although learning based word spotting approaches outperformed other techniques but the necessity of training for every different types of scripts, glyphs, training time and resources (annotated data of properly segmented lines or words, computational resource etc.) are the main impediment of learning based techniques. Although segmentation free approaches provide a solution for these kinds of issues by proposing techniques to spot words without any level of (word level or line level) segmentations, the low accuracy of these techniques are the main drawbacks of segmentation free approaches. So, segmentation based word spotting systems are convenient alternative for word spotting. Although the main problem of such system is it's dependency on segmentation results. If proper segmentation can be attained, segmentation based word spotting systems can give satisfactory results. Moreover, in later chapters, we propose a word spotting technique by introducing new sequence matching techniques, which can handle (up-to certain extend) segmentation errors and also able to work on segmented lines and pseudo words (piece of lines or improperly segmented words).

Chapter 3

Comparative Study of Conventional Time Series Matching Techniques for Word Spotting

Contents

Abstract

In word spotting literature, classical DTW has been widely employed. However there exists several other improved versions of DTW along with other robust sequence matching techniques. Very few of them have been studied in the context of word spotting and this scarcity of research work is the motivation of this chapter. This chapter presents a comparative study of classical Dynamic Time Warping (DTW) technique and many of its improved modifications, as well as other sequence matching techniques in the context of word spotting. An experimental study on historical documents is performed to evaluate

the behavior of DTW's variants and other sequence matching techniques. The comparative analysis shows that classical DTW remains a good choice when there are no segmentation problems and when features are very local. In case of word segmentation errors, Continuous Dynamic Programming (CDP) seems to be a better choice. This research work has also highlights some other sequence matching algorithms in the context of word spotting, which shows interesting results.

3.1 Introduction

Due to the growing utility from different fields of application (signal processing, medical studies, weather forecasting, financial market, etc.) a notable research effort has been devoted to time series matching techniques. Among these techniques, frequently applied ones are Dynamic Time Warping (DTW) and its different variants [Ratanamahatana and Keogh, 2004b, Keogh and Pazzani, 2000, Mayer and Zinke, 2006]. DTW has also been used for word spotting in handwritten manuscripts and historical printed document images. Word images can be thought as 2D signals, which can be matched by sequence matching algorithms [Rath and Manmatha, 2006], [Khurshid et al., 2012], [Jawahar et al., 2004]. In other application domains, DTW's variants have been intensively evaluated to demonstrate their interest [Keogh and Pazzani, 2000, Sakoe and Chiba, 1978], but they have not been clearly studied and compared in the case of word spotting. In this chapter, we propose a detailed comparative study of DTW and it's variants for word spotting, using six types of datasets for (handwritten and old printed documents) experimentation [Mondal et al., 2015b].

The remainder of this chapter is organized as follows. All the datasets used for experiments throughout this chapter and the word spotting framework are mentioned in Section 3.2. The baseline of DTW approach and various other DP paths, warping constraints are briefly explained in Section 3.3. The specific techniques to reduce the quadratic time complexity of DTW algorithm are mentioned in Section 3.4. Several other approaches to improve the quality of DTW is mentioned in Section 3.5. Moreover, other dynamic programming based sequence matching approaches, which has shown better performance than classical DTW in several other domains e.g. shape matching, time series signal matching etc. are mentioned in Section 3.7. An approach to parametrically combine different sequence matching techniques to improve the results are given in Section 3.8. Finally, the study is concluded in Section 3.9.

3.2 Dataset Description and Used Experimental Protocol

For performing following experiments, we used the word spotting framework, described in previous chapter. Only matching techniques are changed each time for evaluating it's performance on some specific datasets. Along with that, the features associated with that specific datasets are also changed. Depending on the characteristics of the algorithms, specific datasets and corresponding features are chosen for evaluation. The description of total 6 datasets are given in the following section.

Dataset-1 (GW-15) This dataset is created from the complete George Washington

dataset, by considering 10 better quality pages among 20 pages. Here we consider total 15 query images. The statistics of this dataset is mentioned in Table 3.1. For the details of this dataset, please see Section 2.5.1 of Chapter 2. The column based feature (refer to Section 2.3.2.1 of Chapter 2) is used to evaluate this dataset.

Table 3.1: Statistics of all queries of GW datasets.

N^o	Query Image		N^o
27(1)	Captain	1755	37(0)
17(0)	Regiment	Company	20(1)
15(1)	October	Cumberland	13(1)
33(0)	Orders	December	26(0)
12(0)	Recruits	Fort	22(0)
12(0)	Sergeant	Instructions	21(1)
14(0)	Virginia	Letters	22(0)
15(4)	Winchester		

$\overline{N^o}$ = No. of full (hyphenated) occurrences;

Dataset-2 (CESR-10) The second dataset is created from CESR data. The detailed description of this dataset is provided in Section 2.5.4 of Chapter 2. We selected 10 queries and, as a test set, the pages where the query (or one of its direct derivative) appears (see Table 3.2). For this dataset also, column based features (refer to Section 2.3.2.1 of Chapter 2) are extracted and used to evaluate the performance of sequence matching techniques.

Table 3.2: Statistics of CESR dataset.

$\overline{N^o}$	$\overline{\overline{N^o}}$	Query word		$\overline{\overline{N^o}}$	$\overline{N^o}$
26(0)	21 (779)	victoire	Chrestien	18 (1449)	15(0)
18(0)	17 (1471)	mortel	cheval	21 (1586)	24(0)
20(0)	22 (1361)	liberté	Cicero	23 (1786)	21(0)
27(0)	22 (811)	François	vaisseau	11 (520)	10(0)
21(0)	22 (1442)	parmy	tantost	13 (804)	20(0)

$\overline{N^o}$ = No. of full (hyphenated) occurrences; $\overline{\overline{N^o}}$ = No. of Pages (No. of words)

Dataset-3 (GW-HOG) This dataset is also created from GW manuscripts but here instead of segmented words, we consider properly segmented lines. From these segmented lines, slit style HOG based features are extracted for the matching purpose. Please note that all the data, corresponding feature values and ground truth are provided by the author of [Terasawa and Tanaka, 2009]. Elaborate description about this dataset and used experimental protocol is mentioned in Section 4.4.1.1 of Chapter 4, which is dedicated to the experiments with subsequence matching techniques.

Dataset-4 (Japanese-HOG) This dataset is a historical Japanese handwritten script having total 1576 segmented lines. There are 4 query words used to evaluate different algorithms applied on this dataset. Please note that we used here the segmented lines, HOG features and Ground Truth, provided by the author of [Terasawa and Tanaka, 2009]. For detail on this dataset, please see Section 4.4.1.2 of Chapter 4.

Dataset-5 (GW-90) and Dataset-6 (Bentham) This dataset is also created from GW dataset. It consists of 4860 segmented words, which are extracted from 20 pages. To choose the queries, initially we grouped all the same words in individual clusters (total 1211

clusters exists i.e. total 1211 unique words are present. Please note that, we ignored "," "-" "." for clustering the words). Now we only choose those clusters of words, which has more than 3 characters and at least 10 elements in their respective clusters. By maintaining aforementioned conditions, we obtain 45 clusters. Now, one query image is randomly chosen from each of the 45 clusters. This process of choosing queries is repeated another time by again randomly choosing one query image from each cluster. So, for this dataset, total $45 \times 2 = 90$ query images were selected as the evaluation set.

The Dataset-6 is created from Bentham collection. For more details about this dataset, please see Section 2.5.3 of Chapter 2. In this case, total 95 queries and 3234 segmented words are provided with the dataset. For the case of Bentham dataset, total 95 queries and 3234 segmented words are provided in the dataset.

Pruning Technique: Please note that, for the case of Dataset-5 and Dataset-6, one pruning step is added with the previously mentioned word spotting framework (refer to Section 2.3 of Chapter 2), for speeding-up the experimental process. Before matching, irrelevant images with respect to the query are discarded by using simple properties of images. The intension is to discard those target images, which are highly different (basic property wise e.g. area, aspect ratio etc.) from the query/reference image. Fine tuning of threshold is highly avoided and we tried to use single, primitive and logical threshold values. After examining several pruning techniques, the following way of pruning is considered, by maintaining the trade-off between useful pruning rate and relaxed/loosely tuned threshold. The objective was to make the system, easily adaptable for other datasets, without much bothering about choosing perfect threshold. Thus, the following constrain was used to determine whether a given word is relevant for a particular query or not:

$$\frac{1}{2} \geq \frac{query\ aspect\ ratio}{target\ aspect\ ratio} \geq 2 \quad (3.1) \qquad \frac{1}{2} \geq \frac{query\ area}{target\ area} \geq 2 \quad (3.2) \qquad \frac{1}{2} \geq \frac{\mathscr{K}\text{ of query}}{\mathscr{K}\text{ of target}} \geq 2 \quad (3.3)$$

Pruning by Eq^n 3.1 does not provide high pruning rate. Due to handwriting variability, words belonging to the same category can often have quite different lengths and areas. For this reason, two other pruning techniques are used. One is based on area of the bounding boxes and the other one is a rough estimation of number of characters in the word. The constraint on area, asserts that words in a cluster have similar areas, the second one asserts that they would contains nearly same amounts of characters. The assumption of bounding box area is obviously does not hold true if words in the same cluster occur at different scales, yet this condition has good pruning capabilities without much lowering the recall, but this technique misses some relevant words, which are of different scales. Whereas the second technique, based on number of characters is an intuitive idea for pruning. This pruning approach is unaffected by the problem of different scaling and multi writer issues. We used a simple technique [Choudhary et al., 2013] for estimating number of characters ($\mathscr{K}$), present in a word. By observing and experimenting the pros and cons of both the pruning approaches separately, it was observed that instead of using these technique separately or sequentially, combining these two criterion together would provide more robustness and high recall to the system. Hence all target images which satisfy the condition-1 (Eq^n 3.1) and condition-2 (Eq^n 3.2) or condition-3 (Eq^n 3.3) are considered as a relevant images.

Although this pruning technique is unable to retrieve all relevant words for every query (it misses only few relevant words), it is a good trade-off between simplicity and high pruning rate.

A well known performance measure, called as precision P_i and recall R_i metrics were used to evaluate the performance of following mentioned sequence matching techniques as well as other algorithms, mentioned in later chapters. Precision P_i is defined as the number of retrieved relevant word instances divided by index i, while recall R_i is defined as the number of relevant word instances divided by the total number of existing relevant words in the dataset. In a typical retrieval scenario, precision is high at the top ranked positions and diminishes gradually while recall behaves in reverse way. The average precision ($AveP$) of retrieving one element is defined as:

$$AveP = \frac{\sum_{i=1}^{\kappa}(P_i \times rel_i)}{\text{number of relevant words}}; \kappa = \text{number of required ranks} \tag{3.4}$$

The rel_i is an indicator function equaling to 1 if the item at rank i is a relevant word, zero otherwise. The mean average precision (mAP) for a set of queries is obtained by calculating the mean of the average precision scores for each query.

$$mAP = \frac{\sum_{q=1}^{Q} AveP(q)}{Q}; Q = \text{total number of queries} \tag{3.5}$$

3.3 Dynamic Time Warping

DTW is a technique for measuring similarity between two different time series by finding their best correspondence. The best warping path, found in a path cost matrix is used to construct the minimum distance between the two time series. Let's assume, two time series $X = x_1, x_2, x_3, ..., x_p$ and $Y = y_1, y_2, y_3, ..., y_q$. To align these two sequences using DTW, we construct an $p \times q$ matrix where the (i^{th}, j^{th}) element of the matrix contains the distance $(\mathfrak{D}(x_i, y_j))$ between two points x_i and y_j (i.e. $\mathfrak{D}(x_i, y_j) = (x_i - y_j)^2$). Each matrix element (i, j) corresponds to the alignment between the points x_i and y_j. The best warping path (W) between these sequences, is a contiguous (described below) set of matrix elements, which defines an optimal mapping between X and Y. The k^{th} element of W is defined as $w_k = (i, j)_k$. The optimal warping path can be defined in following manner:

$$W = w_1, w_2, ..., w_K; \quad max(p, q) \leq K \leq p + q - 1 \tag{3.6}$$

This warping path follow several constraints:

i) **Boundary conditions:** $w_1 = (1, 1)$ and $w_K = (p, q)$. The boundary condition

restrict the warping path to start and finish in diagonally opposite corner cells of the matrix.

ii) **Continuity** The warping path is always continuous, i.e. if $w_k = (m, n)$ and $w_{k-1} = (u, v)$ then $m - u \leq 1$ and $n - v \leq 1$. This restricts the warping path to always go through adjacent cells (including diagonally adjacent cells).

iii) **Monotonicity** The warping path is always monotonically spaced in time i.e. if $w_k = (m, n)$ and $w_{k-1} = (u, v)$ then $m - u \geq 0$ and $n - v \geq 0$.

Many other warping path exist which satisfy the above mentioned three constraints. However the goal here is to find the best optimal path that minimizes the warping cost:

$$DTW(X, Y) = min\sqrt{\sum_{k=1}^{K} w_k} \tag{3.7}$$

This optimal path can be found by using dynamic programming, which defines a path cost matrix $(\mathfrak{P}_{i,j})$ in the following way:

$$\mathfrak{P}_{(i,j)} = \mathfrak{D}_{(i,j)} + min \begin{cases} \mathfrak{P}_{(i,j-1)} \\ \mathfrak{P}_{(i-1,j-1)} \\ \mathfrak{P}_{(i-1,j)} \end{cases} \Bigg|_{i=1,2,....,p;j=1,2,...,q} \tag{3.8}$$

$\mathfrak{P}_{(1,1)} = \mathfrak{D}_{(1,1)}; \underset{1<i\leq p}{\mathfrak{P}_{(i,0)}} = \mathfrak{P}_{(i-1,0)} + \mathfrak{D}_{(i,0)}; \underset{1<j\leq q}{\mathfrak{P}_{(0,j)}} = \mathfrak{P}_{(0,j-1)} + \mathfrak{D}_{(0,j)};$. The optimal distance $(\mathfrak{V})$ is stored in the cell $\mathfrak{P}_{(p,q)}$. For a detailed discussion on classical DTW and some of it's variants, which have been applied in time series matching problems, please see [Albrecht, 2009, Keogh and Pazzani, 2000].

3.3.1 DTW with varying step size condition

In the case of classical DTW, the warping path can get stuck at some position with respect to one sequence, corresponding to local similarity or dissimilarity. To avoid such situation, other step size conditions of DP-path have been proposed in the literature [Sakoe and Chiba, 1978] (see Fig. 3.3)[1]. The symmetric equations are formed by adding some weights on the DP paths to favor vertical, horizontal or $(w_d, w_h, w_v \in R^3)$. The equally weighted case $(w_d, w_h, w_v = 1, 1, 1)$ reduces to classical DTW and $(w_d, w_h, w_v = 2, 1, 1)$ will influence the warping path to take the horizontal and vertical direction. The weighted DP paths are designated as symmetric (red and blue colored text in Table. 3.3) whereas equally weighted DP paths (blue colored text in Table. 3.3) are designated as asymmetric DP paths. Depending on data, these different DP paths help to improve performance of DTW matching over classical DTW approach.

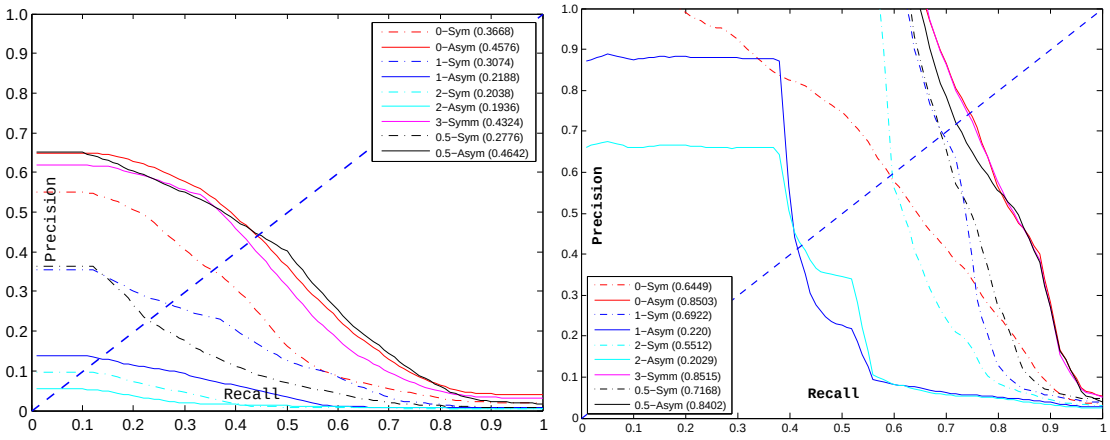
[1] In the case of DP-path 3, only 3-Symmetric (http://www.ee.columbia.edu/ln/labrosa/matlab/dtw/dp2.m) is documented in the literature.

Table 3.3: Symmetric and Asymmetric DP algorithms with various slope constraints. The texts in blue color represents the asymmetric equation of the various DP paths while the text in red color is added with the one in blue, it represents the equations of symmetric DP paths.

Schematic Explanation	**DP Equation**
<u>0</u> 	$\min \begin{pmatrix} \mathfrak{P}(t,\tau-1)+\mathfrak{D}(t,\tau) \\ \mathfrak{P}(t-1,\tau-1)+2\times\mathfrak{D}(t,\tau) \\ \mathfrak{P}(t-1,\tau)+\mathfrak{D}(t,\tau) \end{pmatrix}$
<u>1/2</u> 	$\min \begin{pmatrix} \mathfrak{P}(t-2,\tau-1)+2\times\mathfrak{D}(t-1,\tau)+\mathfrak{D}(t,\tau) \\ \mathfrak{P}(t-1,\tau-1)+3\times\mathfrak{D}(t,\tau) \\ \mathfrak{P}(t-1,\tau-2)+\mathfrak{D}(t,\tau-1)+\mathfrak{D}(t,\tau) \end{pmatrix}$
1 	$\min \begin{pmatrix} \mathfrak{P}(t-1,\tau-2)+[2\times\mathfrak{D}(t,\tau-1)+\mathfrak{D}(t,\tau)] \\ \mathfrak{P}(t-1,\tau-1)+2\times\mathfrak{D}(t,\tau) \\ \mathfrak{P}(t-2,\tau-1)+2\times\mathfrak{D}(t-1,\tau)+\mathfrak{D}(t,\tau) \end{pmatrix}$
2 	$\min \begin{pmatrix} \mathfrak{P}(t-2,\tau-1)+2\times\mathfrak{D}(t-1,\tau)+\mathfrak{D}(t,\tau) \\ \mathfrak{P}(t-1,\tau-1)+3\times\mathfrak{D}(t,\tau) \\ \mathfrak{P}(t-1,\tau-2)+\mathfrak{D}(t,\tau-1)+\mathfrak{D}(t,\tau) \end{pmatrix}$
3 	$\min \begin{pmatrix} \mathfrak{P}(t-2,\tau-1)+2\times\mathfrak{D}(t-1,\tau)+\mathfrak{D}(t,\tau) \\ \mathfrak{P}(t-1,\tau-1)+3\times\mathfrak{D}(t,\tau) \\ \mathfrak{P}(t-1,\tau-2)+\mathfrak{D}(t,\tau-1)+\mathfrak{D}(t,\tau) \end{pmatrix}$

3.3.2 Experimental Protocol and Results

The performance of different DP paths, are investigated through the experiments with different datasets. The results on Dataset-1 and Dataset-2 are shown in Fig. 3.1. It is visible from the curves that the best performing algorithms are; 0-Asymmetric (classical DTW), 0.5-Asymmetric and 3-Symmetric; among all the mentioned DP-Paths. The performance of these three techniques are almost same in both datasets, with small differences only. Most probable reason for this performance is the weights and particular DP-paths, which are comparatively more suitable than others for handling degradation effects and presence of noise in these two datasets. If we compare the results in Dataset-1 with Dataset-2, it is visible that accuracies are significantly higher in Dataset-2. This is mainly due to high degradation effects and the nature of scripts of Dataset-1 (handwritten), whereas in Dataset-2, the presence of noise is comparatively low and features are more distinctive.



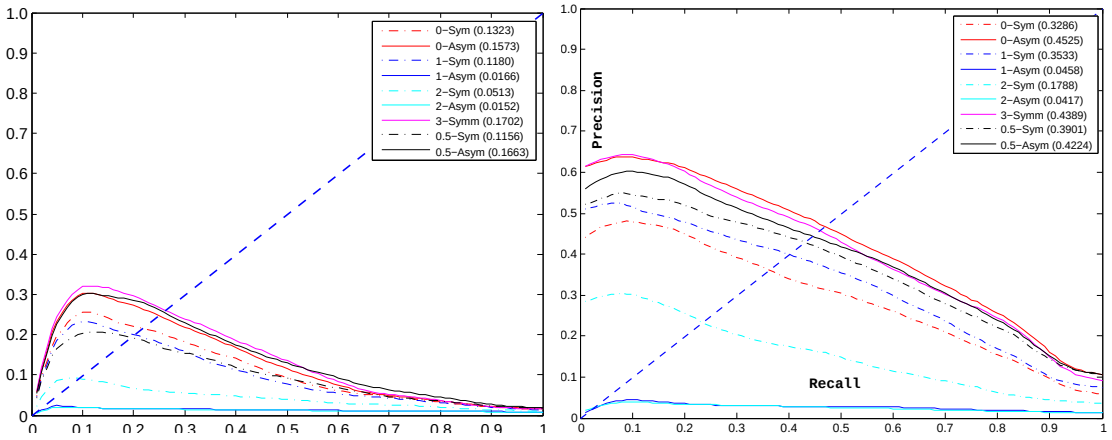
(a) (b)

Figure 3.1: a) Performance comparison of different DP paths on Dataset-1. b) Performance comparison of different DP paths on Dataset-2.

When these DP-paths are applied on comparatively large datasets, i.e. Dataset-5 and Dataset-6, almost similar behavior is visible i.e. 0-Asymmetric (classical DTW), 0.5-Asymmetric and 3-Symmetric have outperformed others (see Fig. 3.2). Although there are small differences in performance between these three techniques but they are not significant. We think that these DP-paths and the corresponding weights are suitable for word image matching in comparison with others. In both cases, the asymmetric DP paths (0-Asymmetric, 0.5-Asymmetric) has significantly outperformed corresponding symmetric ones, which implies that adding weights for favoring any directions of a DP path, is not suitable for finding best warping path for word spotting applications. Whereas giving equal preferences to each direction of a DP path is more helpful. On the contrary to this rationale, 3-Symmetric has also performed well. But please note that, we have not experimented 3-Asymmetric DP-path. As far our knowledge is concerned, 3-Asymmetric DP-path does not exits in literature. The difference in performance between Dataset-5 and Dataset-6 comes from the nature of written scripts and the level of noise present in these datasets.

3.3.3 Global Constraints

Classical DTW technique has high computational complexity. One common way to reduce the computational complexity is to impose global constraint conditions on the admissible warping paths Warping window constraints can be used to restrict the warping path, which intrinsically forces more intuitive alignments, as well as help to speed up the calculation of dissimilarity score between two time series. In case of word spotting, the constrained warping window can also prevents from pathological warping. The most widely used constraints in the literature are Sakoe-Chiba (SC) band and Itakura Parallelogram. Sakoe-Chiba band is a global constraint. It runs along the diagonal and has a fixed (horizontal and vertical) width r. This constraint implies that an element x_i can be aligned only to one of the element y_j with $j \in [\frac{p-r}{q-r}.(q-r), \frac{p-r}{q-r}.(q+r)] \cap [1:p]$. Sakoe-Chiba band can be defined by the following Equation 3.9:



(a) (b)

Figure 3.2: a) Performance comparison of different DP paths on Dataset-5. b) Performance comparison of different DP paths on Dataset-6.

$$i - r \leq j \leq i + r; \quad 1 \leq i \leq p; \text{and } 1 \leq j \leq q. \tag{3.9}$$

The size of the SC-Band can be varied by changing the value of r. The performance of matching highly depends on the chosen value of r. Tuning this value is highly onerous and data dependent.

Moreover, experimental evaluations have shown that the performance of SC-Band and Itakura parallelogram [Keogh and Pazzani, 2000] are similar. Finally, Itakura parallelogram is more suitable choice because it is formed by local slope restrictions, it is a global constraint and there are no tunable parameters. The pseudo code for Itakura parallelogram (DTW) is presented in Algorithm 3 (Procedure ITAKURA). Before going into the

Algorithm 2: ITAKURA PARALLEOGRAM

Input: $i(i \in [1, ...p]), j(j \in [1, ..., q]), p(\text{length of query}), q(\text{length of target})$
Output: *bool* (output is 1 if i and j are within Itakura window, 0 otherwise)

```
bool = 0;
if {j < 2} × {i & i ≤ 2 × j} & {i ≥ (p − 1) − (2 × (q − j))} &
   {j > (q − 1) − (2 × (p − i))} then
    bool = 1;
return bool
```

discussion of another sequence matching technique, we would like to discuss here about one important bottleneck of *Itakura parallelogram*, which has a big impact in our domain of word spotting. By analyzing the mentioned pseudo code of Itakura parallelogram, it is visible that when $q \geq 2 \times p$, *Itakura parallelogram* could not be formed. We consider a toy example to demonstrate the problem. As shown in Fig.3.3, when the length of target signal

is gradually increased, the shape of Itakura parallelogram gets changed and it gradually becomes thinner. The most appropriate shape of Itakura parallelogram can be visible in the left most figure (see Fig. 3.3a), where $p = 200$ and $q = 200$, i.e. when the two sequence have the same size. But when $p = 200, q = 250$ (see Fig. 3.3b), $p = 200, q = 300$ (see Fig. 3.3c), $p = 200, q = 350$ (see Fig. 3.3c) and $p = 200, q = 399$ (see Fig. 3.3d) respectively, the change in shape of Itakura Parallelogram can be visible. When $p = 200$ and $q = 400$, (see Fig. 3.3e) i.e. when the target's length is double of query's length, we could not formulate Itakura parallelogram. This dependence of Itakura parallelogram formation on the length of query and target signal is an obvious factor but it has not been well mentioned in the literature [Ratanamahatana and Keogh, 2004a]. Moreover, no concrete solution is provided in the literature to overcome this problem, except re-sampling. In the experimental section, we have given a simple tricks to handle this problem for word spotting.

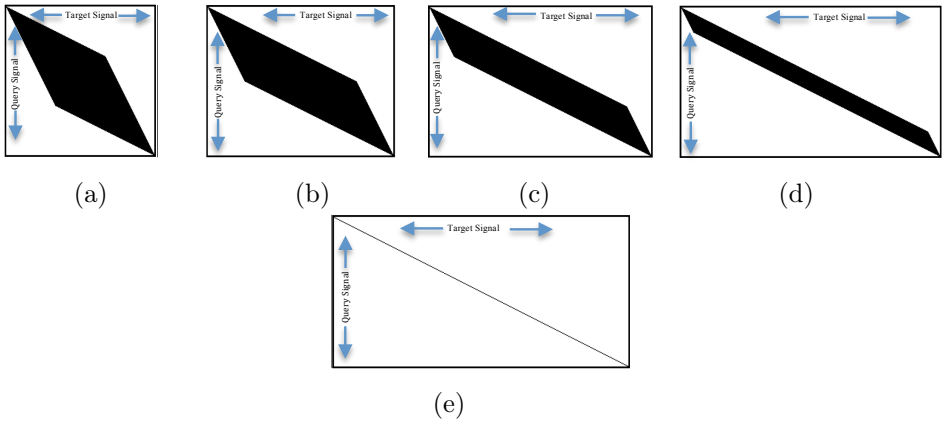


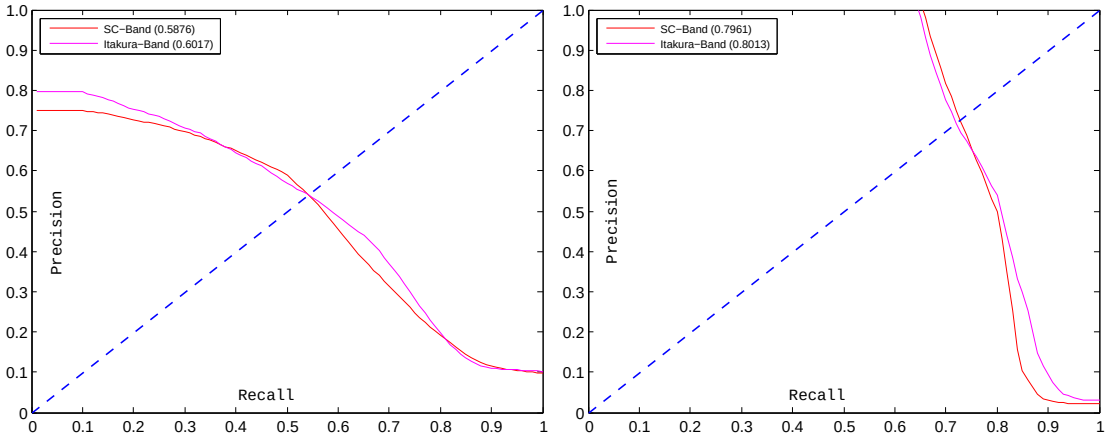
Figure 3.3: The change in the shape of Itakura parallelogram with the change in length of target signal.

For a general constraint region, denoted by R, the path constraint warping path p_R can be computed similar to the unconstrained case by setting $c(x_i, y_j) = \infty$ for all $(i, j) \in [\{1 : N\} \times \{1 : M\}] \setminus R$. So, in the computation of p_R only the cells that lie inside R would be evaluated. Introducing constraints would definitively speedup the process, for example, in the case of Sakeo-Chiba band of a fixed width $T; T << p; T << q$, only $O(T \times max(p, q))$ computations need to be performed instead of $O(pq)$ as required in classical DTW. However the use of global constraints for aligning two sequences could be troublesome, since there is no guarantee that the optimal warping path will definitely go through the specified constraint regions. This fact may lead to undesirable or even completely useless alignment results in some cases but it can also prevent from pathological matching.

3.3.4 Experimental Protocol and Results

To verify the effects of these constraints based DP paths, we experimented it on Dataset-1 and Dataset-2. Fixing the proper radius of SC-Band is a cumbersome task. It takes a lot of manual efforts to find the best performing radius of SC-Band. We have heuristically set the optimized radius as 23% (resp. 30%) of the length of target sequence for Dataset-1

(resp. Dataset-2). The PR plot on Dataset-1 Dataset-2 shows that both constraints have performed equally well whereas Itakura Band has slightly outperformed SC-Band. But the most interesting observation here is that for the case of Dataset-1, we see that constraining the DP path significantly improves the result over classical DTW (0-Asymmetric). The mAP value of SC-Band and Itakura Parallelogram are 0.5876 and 0.6017 for Dataset-1 whereas mAP of classical DTW was 0.4576. So, we can see almost 15% increase in accuracy. But for Dataset-2, this trend cannot be observed. Indeed here, classical DTW has outperformed constrained DTW. The most probable reason of these kinds of behavior is that for Dataset-1, constraining the warping path is helpful as it limits pathological matching which inherently helps it to find best correspondences between query and target sequences, which could vary because of the nature of handwriting in the dataset. On the contrary, completely opposite kinds of behavior can be seen on Dataset-2. Most probably, due to the printed nature of this dataset, constraining the warping path does not help much to find the optimal correspondences and it actually generates resistance to obtain the best warping path. Another possible reason could be our expectation of finding derivatives of queries in Dataset-2, which we have considered as ground truth.



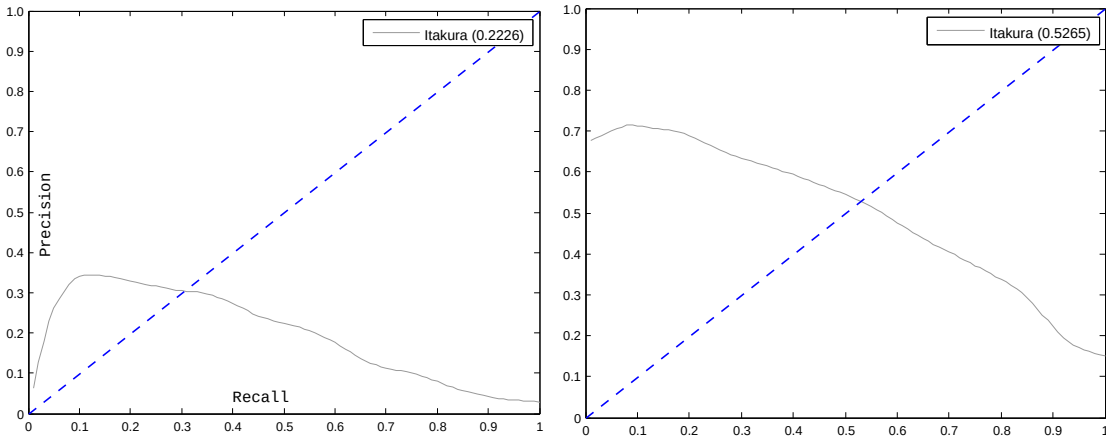
(a) (b)

Figure 3.4: a) Performance comparison of DTW constraints on Dataset-1. b) Performance comparison of DTW constraints on Dataset-2.

Due to the complexity for choosing optimal r, and because previous results shows that SC-Band and Itakura parallelogram has almost similar performance, we did not use SC-Band for our experiments with Dataset-5 and Dataset-6. The performance of Itakura Paralleogam on Dataset-5 and Dataset-6, can be seen from Fig. 3.5a and Fig. 3.5b respectively. The same trend of improvement (i.e. Itakura Parallelogram has outperformed classical DTW on both datasets) can be visible from these two datasets also, which are also handwritten texts as Dataset-1;.

3.4 Speeding up DTW

There are some techniques to reduce the time and space complexity of DTW ($O(mn)$). The methods used to make it faster can be broadly classified into three principal categories.



(a) (b)

Figure 3.5: a) Performance comparison of DTW constraints on Dataset-5. b) Performance comparison of DTW constraints on Dataset-6.

3.4.1 Data abstraction Based

An effective strategy to quickly calculate the DTW distance is by performing the alignment on coarsened versions of the sequences X and Y, thus reducing the lengths of the two considered sequences. Such a strategy is also known as dimensionality reduction or data abstraction. In the following section, we present several of such data abstraction based speedup techniques.

3.4.1.1 Piecewise DTW (PDTW)

Most time series data can be efficiently approximated by piecewise aggregates, so that DTW can operate on higher level of data [Keogh and Pazzani, 2000]. The main idea of PDTW is to reduce the size of original signal by keeping as much information as possible into the reduced signal. Let $\mathfrak{R}$ be the reduced dimension of transformed time series ($1 \leq \mathfrak{R} \leq p$). Although it is not the requirement of the approach but for convenience of explanation we assume that $\mathfrak{R}$ is a factor of p. So a time series X of length p is represented by a vector of $\hat{X} = \hat{x}_1, \hat{x}_2, ..., \hat{x}_{\mathfrak{R}}$. The i^{th} element of $\hat{X}$ is calculated by the equation: $\hat{x}_i = \frac{\mathfrak{R}}{p} \sum_{j=\frac{p}{\mathfrak{R}}(i-1)+1}^{\frac{p}{\mathfrak{R}}i} x_j$. Simply stated, to reduce the data from p dimensions to $\mathfrak{R}$ dimensions, the data is divided into $\mathfrak{R}$ equal sized *"frames"*. The mean value of the data falling within a frame is calculated and a vector of these values becomes the data reduced representation (also called as sub-sampling). Two special cases worth noting are when $\mathfrak{R} = p$ the transformed representation is identical to the original representation. When $\mathfrak{R} = 1$ the transformed representation is simply the mean of the original sequence. More generally the transformation produces a piecewise constant approximation of the original sequence, this approach is named as *Piecewise Aggregate Approximation (PAA)*. We denote the ratio of the length of the original time series to the length of its PAA representation, the compression rate $\mathfrak{c} = \frac{p}{\mathfrak{R}}$. In choosing a value for $\mathfrak{c}$, there is a classic trade-off between memory savings and fidelity. In this work we do not address the problem of choosing the

"best" compression rate. The *"best"* compression rate depends on the structure of the data itself and the task at hand (i.e. clustering/classification/retrieval etc). For most applications the best approach may be to have an expert interact with the data and choose this parameter, although automated approaches to similar problems have been suggested. So this approach of dimensionality reduction is applied on both the query (X) and target (Y) signal. After obtaining the reduced signal the same approach of DTW is applied on $\hat{X}$ and $\hat{Y}$.

3.4.1.2 Fast-DTW

One way to speedup DTW process is to use constraint, where the optimal warping path is calculated thorough the constraint window. Another way to speedup DTW is to use data abstraction. In data abstraction based approach, the DTW algorithm operates on reduced representation of data and the warping path becomes more and more inaccurate as the level of abstraction increases [Salvador and Chan, 2007]. After finding the warping path on reduced representation of data, generally this one is projected on the higher dimension of data. But directly projecting the low resolution warping path to the full resolution, usually creates a warp path, which is far away from the optimal. This is mainly due to the fact that these sorts of projection from reduced representation to higher representation, ignores local variations that could be significant. Fast-DTW is based on both constraints as well as abstraction based approach. Through data abstraction, Fast-DTW first finds the optimal path through a coarse representation of data and then it is refined to the original data, using following mentioned three stages.

Coarsening: Generate reduced representation of the time series that represents the same curve as accurately as possible with less data points.

Projection: After obtaining the reduced representation of the data, the minimum distance warping path at the lower resolution is calculated and used as an initial guess for high resolution's minimum distance warp path.

Refinement: Refine the warping path, projected from lower resolution into the warping path of higher resolution through local adjustment of the warping path.

It was demonstrated in [Salvador and Chan, 2007] that fast DTW runs in $O(N)$ time with sufficient accuracy, and thus this approach speeds up the classical DTW method significantly.

3.4.1.3 Sparse DTW

To compute Dynamic Time Warping (DTW) distance between two time series that always yields the optimal result, a new space-efficient approach (SparseDTW) is proposed in [Al-Naymat et al., 2009]. This technique presents a new approach, which is in contrast to other known approaches which typically sacrifice optimality to attain space efficiency. This technique dynamically exploits the existence of similarity and/or correlation between the time series. The amount of similarity between two time series is proportional to the amount of space required for computing DTW between them i.e. if more similarity exits between the time series the less space required to compute DTW between them. The

convenient way to speedup DTW is to impose priori constraints that does not exploit similarity characteristics that may be present in the data.

3.4.1.4 Accurate and Fast DTW (AF_DTW)

Accurate and Fast DTW (AF_DTW) is an improved version of classical DTW, which is computed by using backward strategy [Li and Yang, 2013]. In contrast to classical DTW, AF_DTW starts from (p, q) to $(1, 1)$ and each element in cost matrix $\mathfrak{P}$ is calculated by considering the maximum of three right adjacent elements subtracting the distance $\mathfrak{D}(i, j)$.

$$\underset{i=p,p-1,....,1;j=q,q-1,.....,1}{\mathfrak{P}_{(i,j)}} = max \begin{cases} \mathfrak{P}_{(i,j+1)} \\ \mathfrak{P}_{(i+1,j+1)} \\ \mathfrak{P}_{(i+1,j)} \end{cases} - \mathfrak{D}_{(i,j)} \tag{3.10}$$
$$\mathfrak{P}_{p+1,q+1} = 0; \mathfrak{P}_{p,q+1} = \mathfrak{P}_{p+1,q} = -\infty$$

After forming the path cost matrix by Eq^n 3.10, the warping path ($P' = \{p_1, p_2, ..., p_K\}$) is calculated in the same way as DTW. The constructed warping path also follows three constraints : boundary condition, continuity and monotonicity. Obviously, this dynamic programming based warping path is used to calculate the best warping path:

$$BS\_DTW(X, Y) = \underset{P}{max} \sum_{k=1}^{K} \mathfrak{D}(p_K) \tag{3.11}$$

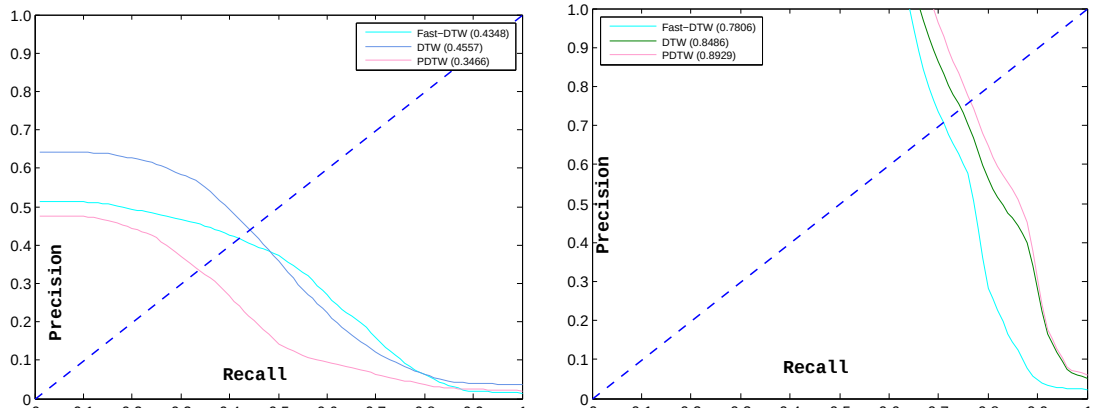
BS_DTW has the same accuracy as DTW and the absolute distance value of BS_DTW (|BS_DTW|) is same as the distance value of DTW. Moreover the time and space complexity of DTW and BS_DTW are same.

In addition, the time and space complexity of BS_DTW is reduced by adopting a strategy, inspired by Itakura and Sakeo-Chiba band. The values in the path cost matrix depends on the initial value at $\mathfrak{P}_{(p+1,q+1)}$. If $\mathfrak{P}_{(p+1,q+1)}$ is initially set to zero then all the elements in $\mathfrak{P}$ are negative but if $\mathfrak{P}_{(p+1,q+1)}$ is set to a positive value then some of the elements in the path cost matrix would be positive and these positive elements would be adjacent. It is shown by some toy example in [Li and Yang, 2013] that the best warping path would always appear in the scope of these positive cells. Let's say $\mathfrak{P}_{(p+1,q+1)} = \theta$ as the initial value. It is shown in the paper that the best warping path always exists in the reduced scope with regard to the special value of θ. So, if the algorithm is forced to find the best warping path in the reduced scope, i.e. only through the positive cells, in such a way the time and space complexity would be reduced. The main problem is to choose a good value of θ so that the best warping path could always be surrounded by the positive cells. It is shown in [Li and Yang, 2013] that a good approximation of θ is $\theta = \sum_{k=1}^{n} (x_p - y_p)^2; n = p = q$

3.4.2 Experimental Protocol and Results

To evaluate the above mentioned approaches for speeding up DTW process, we experimented them on Dataset-1 and Dataset-2. Due to the inherent architecture of Fast-DTW, the accuracy of it would be always lower than classical DTW. But as this method works on coarse representation of data, it will always speedup DTW calculation process with little sacrifice of accuracy. From the following curves, mentioned in Fig. 3.6, it can be seen that in both the datasets, Fast-DTW has low accuracy than the classical DTW but it takes almost half computational time than classical DTW (refer to Table. 3.6). Whereas, along with the reduction of computational time (refer to Table. 3.6), in some cases PDTW can outperform classical DTW. Such an example can be seen in Fig. 3.6b. It can be visible from the experiment on Dataset-2 that PDTW has outperformed classical DTW. The sub-sampling property of PDTW helps it to ignore some local noise and variations, which inherently helps it to improve the results. As PDTW can be considered as a technique to improve the quality of classical DTW, the experimental results of it on Dataset-5 and Dataset-6 are given in the following Section 3.5. As the effectiveness of Fast-DTW (about the reduction in computational time and accuracy) can be visible from small dataset, we have not applied it on larger dataset i.e. Dataset-5 and Dataset-6.

We had carefully implemented the code of Sparse-DTW but most probably due to some implementation issues, it does not behave as expected. Although all other claims of this approach works fine but it could not speedup the DTW calculation process on toy examples. Due to this problem, currently, we have not tested it on our word spotting datasets. Regarding AF_DTW, we have not yet implemented it. We would like to work on both of these techniques in near future.



(a) (b)

Figure 3.6: a) Performance comparison of Fast-DTW and PDTW in comparison with classical DTW on Dataset-1. b) Performance comparison of Fast-DTW and PDTW in comparison with classical DTW on Dataset-2.

3.4.3 Indexing based techniques

For very large time series classification or clustering based problems, such kind of techniques introduce a lower bounding functions for reducing the number of times, DTW must be run during the complete process.

Iterative Deepening DTW is one example mentioned in [Chu et al., 2002]. The intuition behind this algorithm is that based on a model, which describes distribution of distance approximation errors, we can calculate the approximation of DTW at increasingly finer levels of representations. Then we can use the stored model to filter out poor matches with some user defined tolerance. Here the user has to define the tolerance value for false dismissals. In case of zero tolerance for false dismissals this technique will behave as classical DTW. However, with an increasing tolerance value the search technique becomes more faster and by pruning more elements.

The algorithm begins by approximating the query and target series (i.e. X and Y) by PAA approximation technique at a dimensionality reduction level of d_l for obtaining the distance ($D_{PDTW}(d_l)$) in reduced dimensional signal. For the depth d_l and the user confidence (or the user defined acceptable tolerance value for false dismissal), the algorithm determines whether Y could be potential object for possible expansion or not. If so, the algorithm computes the distance in higher resolution i.e. the algorithm computes X and Y with a more precise approximation by using a lower dimensionality reduction rate of d_2 for determining $D_{PDTW}(d_2)$. This process is continued until d has reached the maximum depth. At that point the approximation becomes the original data and we calculate the true warping distance D_{DTW} between the sequence X and Y.

This technique is suitable for dataset containing large amount of time series signals, where the signals are of big lengths. This repetitive process of checking two signals by continuously increasing the level of approximation from fine to finer level could be helpful if the length of the signal is huge (approximatively in thousands). For our case of segmented word image matching, the length of the signals are not very large (mostly in hundreds). So this algorithm would not be much useful for our case as it will take comparatively quite a good amount of time during the process of dimensionality reduction by PAA, and repetitive calculation of DTW on dimensionality reduced signals compared to one time complete calculation of DTW on original signal or calculation of DTW on reduced dimensional signal (one time not multiple time) with rightly adjusted reduction rate. Due to these reason, we have not tested this technique on the datasets. Moreover, we performed experiment with a very small dataset and we observed that the aforementioned facts are true that this technique is not useful for word image matching.

3.5 Improving the quality of DTW

In the previous section, we have observed that there are several drawbacks of classical DTW. In this section, we discuss several improvements proposed in the literature for improving the performance of classical DTW.

3.5.1 Derivative Dynamic Time Warping (DDTW)

In typical condition, DTW tries to explain variability in the Y-axis by warping the X-axis, this may lead to unexpected singularities (the alignments between a point of a series with multiple points of the other series) and unintuitive alignments. In order to overcome those weaknesses of DTW, DDTW transforms the original points into the higher level features, which contain the shape information of a sequence. This technique considers the first derivatives of the signal instead of original one [Keogh and Pazzani, 2000], which intrinsically helps to handle the presence of noise in the signal. The first derivative of sequences gives the information about the *shape* of a signal. With DDTW, the distance measure $\mathfrak{D}(x_i, y_j)$ is not euclidean distance but rather the square of the difference of the estimated derivatives of x_i and y_j. The following techniques is used to generate the derivative of a signal X.

$$\bar{x}_i = \frac{(x_i - x_{i-1}) + ((x_{i+1} - x_{i-1})/2)}{2}; 1 < i < p \tag{3.12}$$

This estimate is simply the average of the slope of the line through the point in question and its left neighbor, and the slope of the line through the left neighbor and right neighbor.

3.5.2 Piecewise Derivative Dynamic Time Warping (PDDTW)

Another similar technique, known as *Piecewise Derivative Dynamic Time Warping* (PDDTW) [Zifan et al., 2007], uses a derivative distance measure in order to reduce singularities and extracting higher level features for capturing the benefit of PDTW and DDTW together. This technique takes the advantage of the fact that the time series can be approximated by *piecewise aggregate approximations* and over that it can use the derivative distance measure to reduce singularities hence can be able to extract higher level features. In order to align two sequences X and Y, firstly a reduced dimensional series $\hat{X}$ and $\hat{Y}$ are obtained respectively. The length of this reduced dimensional series are $\frac{p}{\phi} \times \frac{q}{\phi}$, where the term ϕ denotes the sampling frequency for *piecewise aggregate approximations*. Then, the distance matrix $(\mathfrak{D}(\hat{x}_s, \hat{y}_t); 1 \leq s \leq (\frac{p}{\phi}); 1 \leq t \leq (\frac{q}{\phi}))$ between two elements $\hat{x}_s$ and $\hat{y}_t$ is calculated. To calculate this distance matrix, the square of the difference of the estimated derivatives of elements $\hat{x}_s$ and $\hat{y}_t$ is calculated by the Eq^n 3.12.

3.5.3 Non Isometric Transforms Based DTW

In this technique [Górecki and Luczak, 2014], the authors proposed to use non isometric functions other than the derivative, which can be used in a similar manner and can give better results than DDTW. There are three transforms that are popularly used in technical studies: the cosine transform, sine transform and Hilbert transform. All three are non Isometric for the DTW distance measure. For a series $X = \{x_i : i = 1, 2, ..., p\}$, we have a transform $\bar{X} = \{\bar{x}_k : k = 1, 2, ..p\}$

$$\text{Cosine transform:} \qquad \bar{x}_k = \sum_{i=1}^{p} x_i \; cos[\frac{\pi}{p}(i - \frac{1}{2})(k-1)] \tag{3.13}$$

Sine transform:

$$\bar{x}_k = \sum_{i=1}^{p} x_i \; sin[\frac{\pi}{p}(i - \frac{1}{2})(k-1)] \tag{3.14}$$

Hilbert transform:

$$\bar{x}_k = \sum_{\substack{i=1 \\ i \neq k}}^{p} \frac{x_i}{k-1} \tag{3.15}$$

After obtaining any of these transformed signal, DTW technique is applied on the transformed signal for calculating the dissimilarity value between two sequences.

3.5.4 Value Derivative DTW

One of the drawback of classical DTW is that it only considers data points on Y-axis value. On the basis of DDTW, this algorithm [Kulbacki et al., 2002] propose the following way of calculating distance matrix $\mathfrak{D}$. The derivative of query and target signals are calculated by $\bar{x}_i = x_i - x_{i-1}$ and $\bar{y}_i = y_i - y_{i-1}$. After obtaining the derivated query and target signal, the distance matrix ($\mathfrak{D}$) between these signal is calculated by:

$$\mathfrak{D}(x_i, y_j) = \sqrt{(x_i - y_j)^2}.(\bar{x}_i - \bar{y}_j)^2 \tag{3.16}$$

After calculating the above defined distance matrix, the classical DTW technique is applied for calculating the final distance between the query and target sequence.

3.5.5 Weighted Hybrid Derivative Dynamic Time Warping (WHDDTW)

It is mentioned in [Benedikt et al., 2008] that even DDTW is also noise sensitive, thus an improvement is proposed in this technique. The information from raw signal contains useful information, and smoothing the raw signal helps to stabilise the process. Moreover the derivative provides better knowledge, for example the first derivative gives information on speed and the second derivative gives information on accelerations and decelerations. Thus, the distance matrix ($\mathfrak{D}$) is computed in the following manner, where $\mathfrak{D}_{i,j}$ represent distance between i^{th} and j^{th} elements of query and target signal. Whereas $\bar{\bar{\mathfrak{D}}}_{i,j}$ and $\hat{\mathfrak{D}}_{i,j}$ represents the distance between 1^{st} and 2^{nd} order derivative of query and target signal. Final distance between two elements is given by combining these three terms according to Eq^n 3.17.

$$\mathfrak{D}_{i,j}^{WHDDTW} = w_0.\mathfrak{D}_{i,j} + w_1.\bar{\mathfrak{D}}_{i,j} + w_2.\bar{\bar{\mathfrak{D}}}_{i,j} \tag{3.17}$$

The same process as classical DTW next followed for generating the path cost matrix and final distance between query and target signal. Since derivatives are noise sensitive, too high weight values will affect the performance of the algorithm. Thus, the weight values; w_0, w_1 and w_2, should be chosen by keeping into account the Signal-to-Noise ratio and the difference of magnitudes between the signal and its derivatives. In this study, we choose $w_0 = 1$, $w_1 = 2$ and $w_2 = 2$. The complexity of the algorithm is same as DTW.

3.5.6 Local Dynamic Time Warping (LDTW)

DTW algorithm is modified here to perform pseudo-local alignment using some specific DP-paths [Listgarten et al., 2005]. This algorithm applies different DP paths at different location of path cost matrix ($\mathfrak{P}$) for handling stretching and compression of individual points in time series data. This approach extends the DTW algorithm to perform pseudo-local alignments with the Local Dynamic Time Warping (LDTW) algorithm. Based on the location in path cost matrix, the DTW equation is changed in the following manner:

$$\underset{1\leq i\leq p-1;1\leq j\leq q-1}{\mathfrak{P}(i,j)} = \mathfrak{D}(i,j)+min\begin{cases} \mathfrak{P}(i-1,j-1) \\ \mathfrak{P}(i-2,j-1)+\mathfrak{D}(i-1,j)+\frac{1}{3} \\ \mathfrak{P}(i-1,j-2)+\mathfrak{D}(i,j-1)+\frac{1}{3} \\ \mathfrak{P}(i-3,j-1)+\mathfrak{D}(i-2,j)+\mathfrak{D}(i-1,j)+\frac{2}{3} \\ \mathfrak{P}(i-1,j-3)+\mathfrak{D}(i,j-2)+\mathfrak{D}(i,j-1)+\frac{2}{3} \end{cases} \tag{3.18}$$

For the last column the DTW equation is:

$$\underset{1\leq i\leq p-1;j=q}{\mathfrak{P}(i,j)} = min\begin{cases} \mathfrak{P}(i-1,j-1) \\ \mathfrak{P}(i-2,j-1)+\mathfrak{D}(i-1,j) \\ \mathfrak{P}(i-1,j-2)+\mathfrak{D}(i,j-1)+\frac{1}{3} \\ \mathfrak{P}(i-3,j-1)+\mathfrak{D}(i-2,j) \\ \mathfrak{P}(i-1,j-3)+\mathfrak{D}(i,j-2)+\mathfrak{D}(i,j-1)+\frac{2}{3} \end{cases} \tag{3.19}$$

For the last row the DTW equation is:

$$\underset{i=p;1\leq j\leq q-1}{\mathfrak{P}(i,j)} = min\begin{cases} \mathfrak{P}(i-1,j-1) \\ \mathfrak{P}(i-2,j-1)+\mathfrak{D}(i-1,j)+\frac{1}{3} \\ \mathfrak{P}(i-1,j-2)+\mathfrak{D}(i,j-1) \\ \mathfrak{P}(i-1,j-3)+\mathfrak{D}(i,j-2) \\ \mathfrak{P}(i-3,j-1)+\mathfrak{D}(i-2,j)+\mathfrak{D}(i-1,j)+\frac{2}{3} \end{cases} \tag{3.20}$$

and finally for the last row and last column, it combines to

$$\underset{i=p;j=q}{\mathfrak{P}(i,j) = min} \begin{cases} \mathfrak{P}(i-1,j-1), \\ \mathfrak{P}(i-2,j-1) + \mathfrak{D}(i-1,j) \\ \mathfrak{P}(i-1,j-2) + \mathfrak{D}(i,j-1) \\ \mathfrak{P}(i-3,j-1) + \mathfrak{D}(i-2,j) \\ \mathfrak{P}(i-1,j-3) + \mathfrak{D}(i,j-2) \end{cases} \tag{3.21}$$

To find the warping path, we must look down the last row and the last column and find the cell that has the smallest value. That cell is the end of a local alignment, and the warping path can be found by tracing back from that cell until we reach at the first row or column.

3.5.7 Weighted Dynamic Time Warping (WDTW)

The standard DTW calculates the distance of all points with equal penalization of each point regardless of the phase difference. The WDTW algorithm penalizes the points according to the phase difference between target and query element to prevent minimum distance distortion by outliers. The key idea is that, if the phase difference is low, smaller weight is imposed (i.e. less penalty is imposed) because neighboring points are important, otherwise larger weight is imposed. This approach penalizes matching of points with phase difference (difference of indices), in order to prevent distortions caused by outliers [Jeong et al., 2011]. While creating the $p \times q$ path cost matrix, the distance between the two points x_i and y_j is calculated by $\mathfrak{D}_w(x_i, y_j) = \|W_{|i-j|}(x_i - y_j)\|_p$, where $W_{|i-j|}$ is a positive weight value between the two points x_i and y_j. The algorithm implies that when we calculate the distance between the two points x_i in sequence X and y_j in sequence Y, the weight value will be determined based on the phase difference $|i-j|$. In other words, if the two points x_i and y_j are near, smaller weights can be imposed. Thus, the optimal distance between two sequences is defined as the minimum path over all possible paths as follows:

$$WDTW_p(X,Y) = \sqrt[p]{\mathfrak{P}(i,j)} \tag{3.22}$$

where, $\mathfrak{P}(i,j) = |W_{|i-j|}(x_j - y_j)^p| + min\{\mathfrak{P}(i-1,j-1), \mathfrak{P}(i-1,j), \mathfrak{P}(i,j-1)\}$ A technique called as *"Modified logistic weight function(MLWF)"* is used to systematically assign weights as a function of phase difference between two points. The parameter g controls the amount of penalization considering phase difference. The weight value $W_{(i)} = \frac{W_{max}}{1+e^{(-g(i-m_c))}}$ where $i = 1, ..., p$, p is the length of a sequence and p_c is the midpoint of a sequence. W_{max} is the desired upper bound for the weight parameter and g is an empirical constant that controls the curvature of the function; that is g controls the level of penalization for the points with larger phase difference. The value of g could range from zero to infinity.

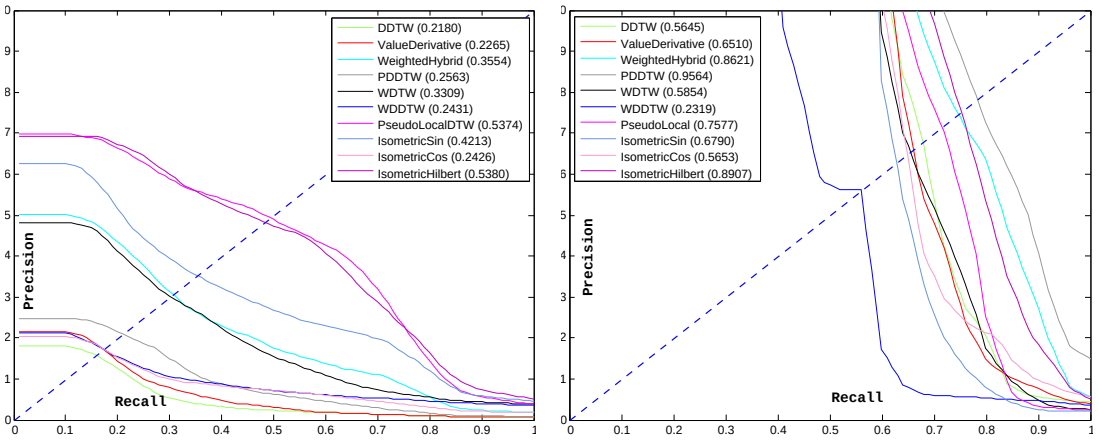
3.5.8 Weighted Derivative Dynamic Time Warping (WDDTW)

The proposed concept of weighted dynamic time warping can be extended to variants of DTW. In this sub-method, the proposed idea of derivative dynamic time warping is extended to weighted version, which is called weighted derivative dynamic time warping.

The weighted version of DDTW (WDDTW) is calculated by using the same Eq^n 3.22 on $\bar{X}$ and $\bar{Y}$ respectively; where $\bar{X}$ and $\bar{Y}$ are the 1^{st} order derivative of query and target signals respectively; i.e. $WDDTW_p(X_i, Y_j) = WDTW_p(\bar{X}_i, \bar{Y}_j)$

3.5.9 Continuous DTW (CDTW)

Here a sample point in one of the signal is allowed to be matched with an implicit point lying between two sample points of the other signal [Munich and Perona, 1999]. This method compares planar curves which enables it to perform matching at sub-sampling resolution. Due to the intrinsic characteristics of this algorithm, it is difficult to properly adapt this technique for word image matching problem. Due to this issue, we have not implemented and applied this approach for our evaluation purpose.



(a) (b)

Figure 3.7: a) Performance comparison of different algorithms for improving the quality of DTW on Dataset-1. b) Performance comparison of different algorithms for improving the quality of DTW on Dataset-2.

3.5.10 Experimental Protocol and Results

To evaluate the performance of these above defined algorithms in this section, we performed experiments with 4 datasets.

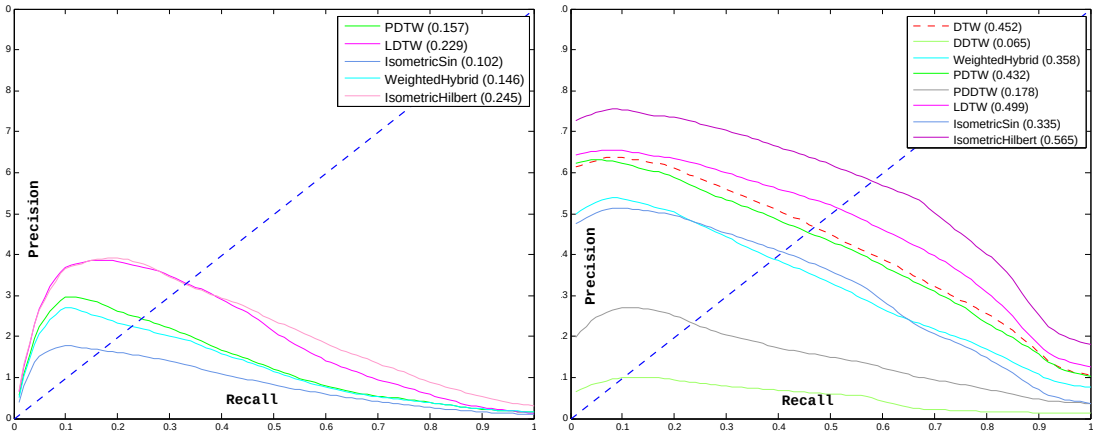
The experiment on Dataset-1 and Dataset-2 shows some improvement of accuracy in comparison to classical DTW. It can also be observed that different algorithms perform differently in the case of handwritten and printed documents. For example, in Dataset-1 (refer to Fig.3.7a) *Isometric Hilbert* has outperformed all other techniques, whereas *Pseudo Local DTW* is second best. Moreover it can also be visible that *Isometric Sin* and *Weighted Hybrid DTW* has also performed good enough in comparison with rest of the algorithms. A different behavior is visible, when these algorithms were applied on machine printed documents (Dataset-2). We can observe (refer to Fig. 3.7b) from the result that PDDTW

has outperformed all others, whereas PDTW has not performed well enough. In this dataset also, *Isometric Hilbert* has outperformed others except PDDTW. The performance of *Weighted Hybrid DTW* is quite competitive also. Another fact here is that although the performance of *Pseudo Local DTW* is one of the best but it does not perform as good as in Dataset-1.

From the analysis of results, it is interesting to observe that Hilbert transform and Sin transform on word image feature values, add distinguishable property to the image features, which is better than typical derivative of image feature values. Moreover, it can also be observed that weighted hybridization of original signal and it's 1^{st} and 2^{nd} order derivative (Weighted Hybrid DTW) is significantly useful for word image matching. Most probably, the variable performance of Pseudo local DTW on this two datasets is due to the presence of more local variations and degradation noise in Dataset-1 than Dataset-2. The results of DDTW v/s DTW, PDDTW v/s PDTW and WDDTW v/s WDTW shows that the local variation and noise can't be handled by simple derivative operation on the signals. Instead of improving the results, these derivative operations actually deteriorate the results. The same pattern of behavior can also be observed from Dataset-2, where the derivative operation on original signals highly reduces the accuracies. The interesting performance of PDDTW on Dataset-2 is mainly due to the sub-sampling of signal derivatives. The same technique i.e. PDDTW has not performed well in Dataset-1 because of it's nature (handwritten texts) and presence of local variations. Weight factors (g) for WDTW (resp. WDDTW) are set at 0.31 (resp. 0.26) for Dataset-1 and at 0.03 (resp. 0.01) for Dataset-2.

After observing interesting behavior of different aforementioned algorithms on Dataset-1 and Dataset-2, we choose some of the best performing techniques for comparatively bigger datasets. So, we decided to experiment the following algorithms on Dataset-5 and Dataset-6: **i)** *Isometric Hilbert DTW* **ii)** *Isometric Sin DTW* **iii)** *Weighted Hybrid DTW* **iv)** *PDTW* **v)** *PDDTW* **vi)** *Pseudo Local DTW*. The result of these techniques can be seen in the following Fig. 3.8. As both of these datasets are consists of handwritten historical texts, we first apply all of these techniques on Dataset-6 (experiments on this dataset is comparatively fast). The P-R curves of the techniques are plotted in Fig. 3.8b. It is visible from the results that the algorithms shows similar kinds of behavior as the ones in Dataset-1 and Dataset-2. Here also Isometric Hilbert has outperformed other techniques, whereas Isometric Sin has also performed well. In the similar manner, weighted hybridization of original and derivative signal, performs better than simple derivative of signals. LDTW has also performed well due to the presence of local variations. Interestingly the performance of PDTW is significantly higher than PDDTW, which implies that sub-sampling of original signal is better than transforming the signal by derivative operation.

After observing the results on Dataset-6, we applied the best performing algorithms on Dataset-5. The P-R curves given in Fig. 3.8a shows that according to our expectation, Isometric Hilbert outperformed all others while LDTW, Weighted Hybrid and PDTW has also performed well. This performance on Dataset-6, justify our rationale, given for Dataset-1 and Dataset-5.



(a) (b)

Figure 3.8: a) Performance comparison of different algorithms for improving the quality of DTW on Dataset-5. b) Performance comparison of different different algorithms for improving the quality of DTW on Dataset-6.

3.6 Finding subsequence with DTW

All of the algorithms, mentioned in above section (refer to Section 3.5) was designed for matching sequences of similar length (approximatively). As all of these technique proposes some modification of classical DTW, so these techniques can be applicable for sequences of different lengths, which classical DTW can handle also. But same as DTW, none of these above mentioned techniques can handle subsequence matching, which is needed for Dataset-3 and Dataset-4. In this section (refer to Section 3.6.1), we have described a simple and useful modification of classical DTW, which can handle subsequence matching and we strongly think all of these above mentioned techniques can be well applicable for subsequence matching. But this proposition has not been explored in this research work. We would like to explore this proposition in near future.

3.6.1 Subsequence DTW (SSDTW)

This algorithm is designed for finding a continuous subsequence within a longer sequence that can optimally fit the shorter query sequence [Albrecht, 2009]. Classical DTW is modified in order to be able to match a sequence with a subsequence of a longer target sequence. It works by relaxing the boundary conditions of classical DTW. Let $X = (x_1, x_2, ...x_p)$ and $Y = (y_1, y_2, ...y_q)$; $q \gg p$ be the sequences of features. The goal is to find the continuous indices from a^* to b^* of Y so that $Y(a^* : b^*) = (y_{a^*}, y_{a^*+1}, ..., y_{b^*})$ with $1 \leq a^* \leq b^* \leq q$ that minimizes the DTW distance to X over all possible subsequences of Y. In other words: $(a^*, b^*) = \underset{(a^*,b^*):1\leq a^*\leq b^*\leq q}{\text{argmin}} (DTW(X, Y(a^* : b^*)))$. These continuous indices of Y can be obtained by violating the boundary condition property of classical DTW, i.e the formation of warping path (backtracking) can start from any

column at the last row of path cost matrix and can end at any column at the first row of path cost matrix. The path cost matrix ($\mathfrak{P}$) is initialized in the following manner. $\mathfrak{P}_{(1,1)} = \mathfrak{D}_{(1,1)}; \underset{1<i\leq p}{\mathfrak{P}_{(i,0)}} = \mathfrak{P}_{(i-1,0)} + \mathfrak{D}_{(i,0)}; \underset{1<j\leq q}{\mathfrak{P}_{(0,j)}} = \mathfrak{P}_{(0,j-1)} + \mathfrak{D}_{(0,j)}$;. The optimal distance ($\mathfrak{V}$) is stored in the cell $\mathfrak{C}_{start} = [\underset{1\leq \mathfrak{t}\leq q}{\text{argmin}}\{\mathfrak{P}_{(p,\mathfrak{t})}\}]$. The backtracking for warping path calculation, also starts from this particular cell, which ends at any cell in 1^{st} row; i.e. at $\mathfrak{C}_{end} = [\underset{1\leq \mathfrak{t}\leq q}{\text{argmin}}\{\mathfrak{P}_{(1,\mathfrak{t})}\}]$.

3.6.2 DTW with Correspondence Window (DTW-CW)

Here a query sequence is matched against a subsequence of a target sequence using a sliding correspondence window (same length as query). So the technique is a sliding window (w) based approach with overlapping of ($w - \zeta$). The width of sliding window (w) in target sequence is typically equal to the length of query sequence. The position of optimal subsequence within the target sequence is obtained by calculating DTW distance between each slided window and the query sequence. Due to this repetitive way (calculation of DTW on each slided window) of calculating best subsequence, DTW-CW is highly computationally expensive.

Choosing the value of ζ could be troublesome for different applications. In the case of word spotting applications, one possibility of automatically choosing ζ as the average width of characters, belonging to reference word. It is meaningful for word spotting problem to skip each time the width equals to average character width in sliding window based matching. Otherwise ζ can be fixed heuristically, based on the properties of the datasets.

3.6.3 Meshesha-Jawahar DTW (MJ-DTW)

A DTW based partial matching technique, dedicated to word spotting is proposed in [Meshesha and Jawahar, 2008b]. It can take care of variations at the beginning and at the end of extracted sequences from segmented words. The correspondence of target and query feature sequences can be concentrated at the end, at the beginning or both, when the target word have pre and/or post conjugation, with respect to query word. Due these sorts of concentration issues, the warping path deviates (from diagonal) in the horizontal or in vertical directions at the beginning or end. It is note-worthy to mention that as the matching path deviates from the diagonal line, the matching cost increases staggeringly. Profiles from these pre and/or post extra characters have no or minimal contribution for measuring similarity of the two words, so logically their cost needs to be cut down from the total matching score. Hence, to reduce unwanted extra cost, this technique first analyze whether the dissimilarity between words is concentrated at the end, at the beginning or both. After performing the analysis, the extra cost at the two extremes are removed to reduce the total matching score and to obtain the optimal dissimilarity value.

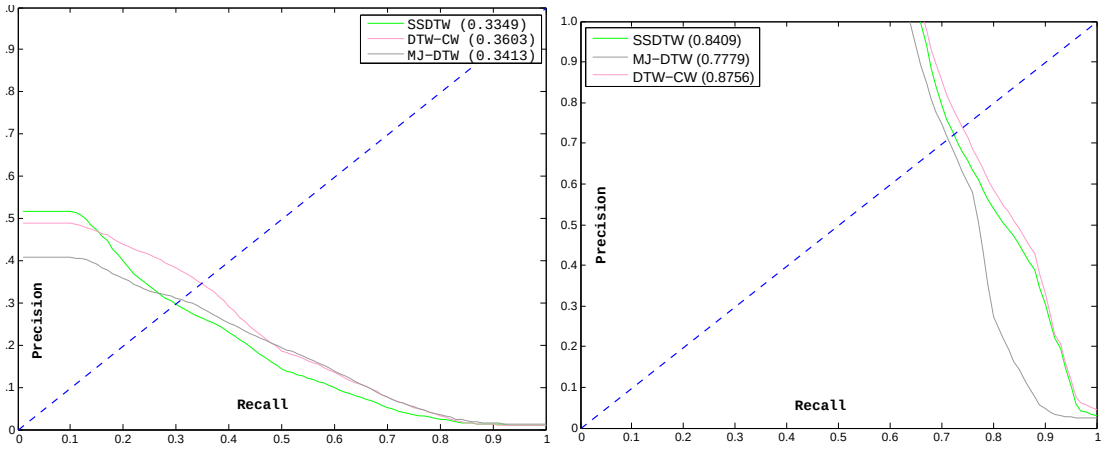
3.6.4 Experimental Protocol and Results

The results of above mentioned three techniques on Dataset-1 and Dataset-2 are shown in Fig. 3.9. Please note that ζ is taken as average character width for Dataset-1 and ($2 \times$ average stroke width) for Dataset-2. Average character width is calculated by obtaining individual characters or glyphs by connected component labeling technique (CCL)[2], whereas average stroke width for handwritten characters is calculated by the technique mentioned in [Chen, 2011]. To match a query word with pseudo words in Dataset-1, it is logical to use average character width as the skipping/overlapping parameter (ζ) for DTW-CW. By this way, it will be able to find approximatively the sequence of characters, which will optimally match with the characters in query image. In the case of Dataset-2, it is highly difficult to have a proper estimation of average character width (because it is handwritten). So, we decided to consider stroke width as an approximation of character width.

It is visible from the plots in Fig. 3.9a and Fig. 3.9b, that comparatively these three algorithms have performed well enough but among them DTW-CW has slightly performed better than other two in both the datasets. Whereas, SSDTW has minutely outperformed MJ-DTW. Although DTW-CW has performed well, but it's high computational cost due to the exhaustive search for obtaining the best match is a bottleneck of this technique (see Section 3.6.2). Moreover deciding the value of ζ is also a cumbersome process. Whereas due to the inherent architecture of SSDTW, it is computationally inexpensive and also able to provide comparative high accuracy. So it can be concluded that in these datasets, the partial sequence matching approaches are more robust compared to the techniques (e.g. MJDTW), which are able to handle pre and/or post conjugation of words. The results on Dataset-1 is significantly lower than the ones in Dataset-2. Which implies that subsequence matching techniques works well on improperly segmented words than properly segmented ones. Likewise other reasons for this difference in accuracies are: presence of noise, degradation effects, nature of the scripts (handwritten v/s machine printed) etc.
From the experiments on Dataset-1 and Dataset-2, we observed that all the three techniques mentioned in this section have performed satisfactorily and almost equally. But to evaluate the performance on the datasets of segmented lines (Dataset-3 and Dataset-4), we only have experimented SSDTW and DTW-CW on these two datasets. Application of MJDTW is irrelevant here as this technique is specifically designed for segmented words, for finding pre and/or post conjugations of a word.

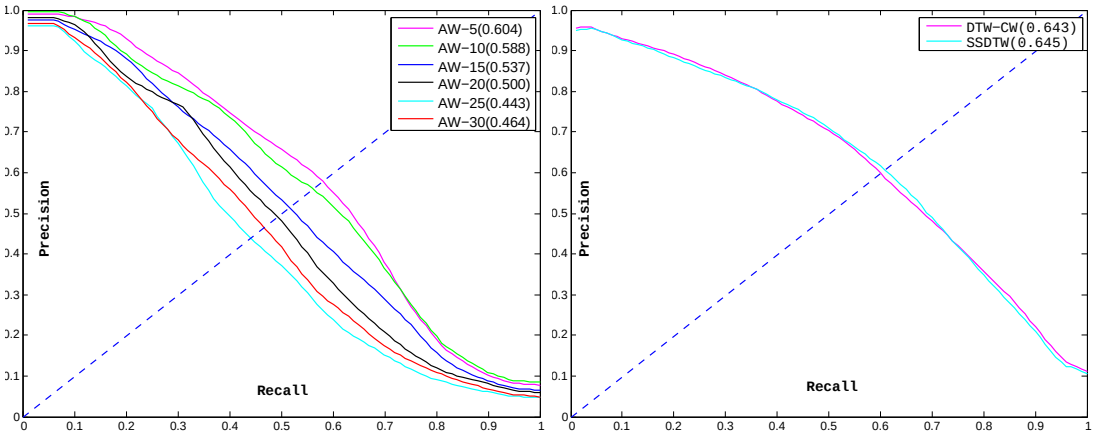
Fixing the skipping parameter for DTW-CW is a tedious task for the case of Dataset-3 and Dataset-4. Due to the complex script structure of Japanese language and historical degradation effect on GW dataset, it is quite difficult to get the average character width, as the segmentation of characters is highly difficult. So, we decided to find this threshold by heuristic manner. We performed the experiments by increasing the value of ζ with a gap of 5, i.e. $\zeta = 1, 5, 10, 15, 20, 25, 30$ are considered. The process of increasing the value of ζ were stopped when the accuracy start decreasing or become stable. Although from the plot in Fig. 3.11a and Fig. 3.10a, it can be visible that the accuracy started decreasing significantly from $\zeta = 10$ but we continue to increase the value of ζ to verify whether any strange behavior could occur or not (it didn't occur). This analysis implies that, more

[2] `https://en.wikipedia.org/wiki/Connected-component_labeling`



(a) (b)

Figure 3.9: a) Performance comparison of DTW-CW technique for different values of sliding width (AW) (see digitized version of the image for better clarity). b) Precision-Recall plot for SSDTW.

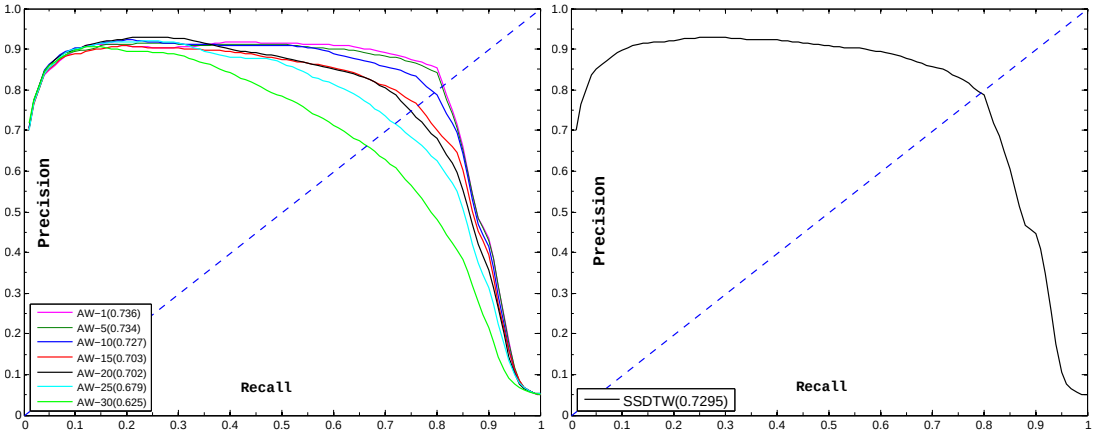


(a) (b)

Figure 3.10: a) Performance comparison of DTW-CW technique for different values of sliding width (AW) for Dataset-3. b) Precision-Recall plot for DTW-CW and SSDTW for Dataset-3.

there is overlapping (ζ), better are the results.

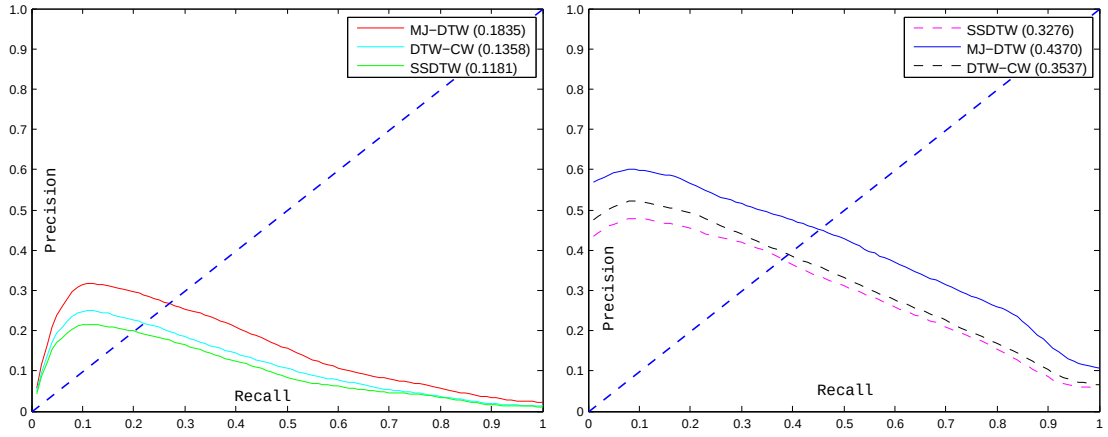
It can be visible that the best result obtained with $\zeta = 1$ but using $\zeta = 1$ is highly time consuming so we consider the value of $\zeta = 5$ as the best trade off between accuracy and speed. Although $\zeta = 10$ could also be considered. The accuracy obtained by DTW-CW ($\epsilon = 5$ & 10) is comparable with the accuracy of SSDTW (minutely less than DTW-CW). However SSDTW is computationally inexpensive in comparison with DTW-CW. Although ζ = stroke width of the characters can also be directly considered here but just to verify the effect of different thresholds, we performed the aforementioned approach. It is



(a) (b)

Figure 3.11: a) Performance comparison of DTW-CW technique for different values of sliding width (AW) for Dataset-4 (see digitized version of the image for better clarity). b) Precision-Recall plot for SSDTW for Dataset-4.

noteworthy to mention here that ideally we should use a separate learning set for obtaining the optimized value of ζ and this optimal value of ζ should be used for testing set. But as we have only 4 queries in dataset-3, it is not feasible to divide these 4 queries into learning and testing set. Moreover, for comparing the result of DTW-CW with other matching techniques, it is necessary to use all 4 queries to evaluate all matching techniques on the same experimental protocol. Due to the same reason, we performed experiment on all 15 queries of GW dataset also. The P-R curves are shown in Fig. 3.10b for Dataset-3 and in Fig. 3.11b for Dataset-4.



(a) (b)

Figure 3.12: a) Performance comparison of different algorithms in this section on Dataset-5. b) Performance comparison of different algorithms in this section on Dataset-6.

From the above shown experiments with Dataset-1 and Dataset-2, we saw that in

Dataset-1 DTW-CW has outperformed other two techniques whereas MJ-DTW has performed better than SSDTW. In the case of Dataset-2, although DTW-CW has again outperformed the other two but in this dataset SSDTW has performed better than MJ-DTW. By analyzing the algorithms on small datasets, we observed that the inter performance difference is not very high and the accuracies are close to each other; i.e. all the three techniques in this section have performed equally well.

After seeing the behavior on smaller datasets, we applied these algorithms on the large dataset i.e. Dataset-5 and Dataset-6. The performance of these algorithms are shown in following Fig. 3.12. It can be seen from these P-R plots that MJ-DTW has outperformed others in both the datasets, whereas DTW-CW is the second best. Please note that for the case of DTW-CW, we consider average stroke width as the parameter for overlapping window size. As mentioned before also that DTW-CW do exhaustive search for finding the optimal sub-sequence so even if other sub-sequence matching techniques has little bit low accuracy but they are significantly computationally inexpensive than DTW-CW. The best performance of MJ-DTW implies that searching pre and post conjugations of query words are helpful for Dataset-5 and Dataset-6.

3.7 Other relevant sequence matching techniques

There are others relevant sequence matching techniques mentioned in the literature. These methods were proposed to overcome some of the architectural drawbacks of DTW by removing some constraints (especially boundary and continuity conditions), which helps these techniques to skip outliers from query and/or target sequences. But on the contrary, the many-to-one and one-to-many matching property of DTW is missing in the following mentioned techniques.

3.7.1 Longest Common Subsequence (LCSS)

The longest common subsequence dissimilarity measure is an algorithm which is proposed on the foundation of *edit distance* or *Levenshtein distance* measure, primarily used in speech recognition domain. The basic idea is to match two sequences by allowing them to stretch, without rearranging the sequence of the elements. LCSS also facilitates some elements to be unmatched or left out (e.g., outliers), whereas in Euclidean Distance and DTW, all elements (even the outliers) from both sequences must be used for obtaining final distance between two sequences. The overall idea is to get the longest common sub-sequence from the query and target sequences. The obtained longest common sub-sequence, does not need to be made of consecutive points, but the order is preserved and some points can be skipped from matching. Unlike DTW, one point can never be associated twice with points of the other sequence, so the maximum number of correspondence is the smaller length of the two sequences. The LCSS [Vlachos et al., 2003] measure has two parameters, δ and ϵ. The constant δ, which is usually set to a percentage of the sequence length, is a constrained warping band to control the window size for matching a given point from one sequence to a point in another sequence. It controls how far in time, we can go in order to match a given point from one trajectory to a point in another trajectory. The constant

$0 < \epsilon < 1$ is the matching threshold: two points from two sequences are considered for matching, if their distance is less than ϵ. As, LCSS was originally proposed for symbolic sequences (i.e. character strings), so to apply it for numeric values, a threshold is required to determine when two close numeric values can be treated as equal or not. The performance of LCSS is highly depends on the correct setting of this threshold, which may be a particularly difficult problem in some applications.

By considering the ratio between the length of the calculated longest common subsequence and the length of the whole sequence, the dissimilarity between query and target sequence is calculated. Since the inherent goal of finding longest common subsequence is to find optimal substructure between two compared sequence, the problem of LCSS is often solved with dynamic programming. Given two sequence $X = x_1, x_2, ...x_p$ and $Y = y_1, y_2, ..., y_q$, the length of their longest common sub-sequence, denoted as: $\mathscr{L}(X, Y); 1 \leq i \leq p$ and $1 \leq j \leq q$; is calculated by:

$$\mathscr{L}_{i,j} = \begin{cases} 0 & if \quad i = 0 \quad and \quad j = 0 \\ \mathscr{L}_{i-1,j-1} + 1 & if \quad |x_i - y_j| < \epsilon \quad and \quad |i - j| \leq \delta \\ max(\mathscr{L}_{i,j-1}, \mathscr{L}_{i-1,j}) & \text{Otherwise} \end{cases} \tag{3.23}$$

As, LCSS allows skipping elements of both the query and the target sequence, therefore LCSS should be used when one is interested in finding the best matching part between the target sequence and a given query sequence, since it guarantees that the whole query sequence will be matched. When the query sequence contains outliers and skipping them is allowed, then LCSS should be used. $\mathscr{L}_{i,j}$ contains the similarity between X and Y, because it corresponds to the length of the longest common subsequence of elements between time series X and Y. To define the dissimilarity between X and Y, we can compute:

$$LCSS(X, Y) = \frac{p + q - 2\mathscr{L}_{p,q}}{p + q} \tag{3.24}$$

According to this definition, this measure takes values from $LCSS(X, Y) = \frac{p+q-2\mathscr{L}_{p,q}}{p+q}$ to 1. For two trajectories of equal length it takes values from 0 to 1. Taking into account only sufficiently similar points, LCSS solves the problem of the presence of noise, but does not satisfy the triangle inequality [Vlachos et al., 2003], so it is not a distance metric. LCSS is robust to noise and is expected to be more accurate than DTW in the presence of outliers.

3.7.2 Derivative based longest common sub-sequence

Let's consider LCSS be the longest common subsequence dissimilarity measure between two time series X and Y.

3.7.2.1 1D-LCSS

Same as DDTW, in this approach, derivative of the signals are considered, instead of the original ones for making LCSS more robust to noise [Górecki, 2014]. It proposes a dissimilarity measure by considering both the function values of time series and values of the first derivative is defined as:

$$1D_{LCSS}(X,Y) = a \times LCSS(X,Y) + b \times LCSS(\bar{X},\bar{Y}) \tag{3.25}$$

where $\bar{X}$ and $\bar{Y}$ are the first discrete derivative of X and Y and $a, b \in [0,1]$ are parameters. The discrete derivative of a time series X of length p is defined by: $\bar{X}(i) = X(i+1) - X(i); i = 1, 2, ..., p-1$

3.7.2.2 DD-LCSS

This is a dissimilarity measure, which considers both the function values of time series and values of the first and second derivative is defined by:

$$DD_{LCSS}(X,Y) = a \times LCSS(X,Y) + b \times LCSS(\bar{X},\bar{Y}) + c \times LCSS(\bar{\bar{X}},\bar{\bar{Y}}) \tag{3.26}$$

Where, $\bar{\bar{X}}$ and $\bar{\bar{Y}}$ are the second derivatives of X and Y and $a, b, c \in [0,1]$ are the parameters [Górecki, 2014]. If the similarity measure in the above definition is a metric then the new measure $1D_{LCSS}$ and DD_{LCSS} are also metrics.

3.7.3 Minimal Variance Matching (MVM)

This algorithm is designed to handle partial sequence matching and also to skip outliers from target sequence. *Minimal Variance Matching* [Latecki et al., 2007b] combines the strengths of both DTW and LCSS. MVM tries to find an optimal path including all query points but it is able to skip outliers from target sequence during the matching process (up to a given limit). MVM provides a relation $\mathbb{R}$ with an injection (1-1 mapping) from X to Y. More precisely, each x_i would map to exactly one y_j, in an order preserving manner, while some elements (considered outliers) would be skipped from matching with query sequence. Due to this outliers skipping property of MVM, the influence of outliers in Y on the dissimilarity value between X and Y can be eliminated. Hence, the query sequence X is matched with only a part of Y, by this way, this method allows to subsequence of Y, which optimally matches with query sequence X and the final dissimilarity value is calculated between Y' and X. The given correspondence by MVM is a monotonic injection.

$$f : \{1, ..., m\} \rightarrow \{1, ..., n\}; \text{ where } f \text{ is a function, such that } f(i) < f(i+1) \tag{3.27}$$

X_i is mapped to $Y_{f(i)}$ for all $i \in \{1, ..., p\}$. The set of indices $f(1), ..., f(m)$ defines a sub-sequence Y' of Y. It is noteworthy here to mention that, in case of DTW, the correspondence is a relation on the set of indices $\{1, ..., p\} \times \{1, ..., q\}$, i.e. a one-to-many and many-to-one mapping. Once the correspondence is known, it is easy to compute the distance between functions.

$$\mathfrak{D}(X, Y, f) = \sqrt{\sum_{i=1}^{m} (Y_{f(i)} - a_i)^2} \tag{3.28}$$

The optimal correspondence $\hat{f}$ of values in series X to values in series Y is obtained by calculating the global minimum of $\mathfrak{D}(X, Y, f)$ over all possible correspondences of $f : [\hat{f} = \arg\min\{\mathfrak{D}(X, Y, f)\} : f \textit{ is a correspondence}]$. Finally the optimal distance is obtained as :

$$\mathfrak{D}(X, Y) = \mathfrak{D}(X, Y, f) = \sqrt{\sum_{i=1}^{m} (Y_{f(i)} - a_i)^2} \tag{3.29}$$

So, the above distance $\mathfrak{D}(X, Y)$ is the global minimum over all possible correspondences.

3.7.4 Optimal Sequence Bijection (OSB)

This algorithm extends the MVM [Latecki et al., 2007c] algorithm. It is particularly suitable for partial and elastic matching as it can skip outliers present in query as well as in target. In this way, OSB can also match a query longer than the target, which was not possible with MVM. The goal of OSB is to find subsequences X' of $X; (X' \in X)$ and Y' of $Y; (Y' \in Y)$ such that X' best matches Y'. Due to the possibility of having outliers inside the sequences X and Y, skipping some elements of X and Y can be very crucial. However, the freedom of skipping elements should also be restricted to prevent unnecessary correspondences. To solve this purpose, a penalty of skipping is introduced, which is called as *"jumpCost"*, denoted as $\mathfrak{C}$. The distance between X and Y are kept inside the distance matrix $\mathfrak{D}$. The optimal correspondence can be found by generating a DAG, using the distance matrix $\mathfrak{D}$. The nodes of the DAG are all index pairs $(i, j) \in \{1...p\} \times \{1...q\}$ and the edge cost $\mathcal{W}$ is defined as:

$$\mathcal{W}\{(i,j)(k,l)\} = \begin{cases} \sqrt{(k-i-1)^2 + (l-j-1)^2}.\mathfrak{C} + \mathfrak{D}(x_k, y_l) & if \quad i < k \bigwedge j < l \\ \infty & \text{otherwise} \end{cases} \tag{3.30}$$

Thus, the cost of an edge from (i, j) to (k, l) is the Euclidean distance of vertices (i, j) and (k, l) in the matrix $\{1...p\} \times \{1...q\}$ times the jump cost plus the dissimilarity measure of

elements x_k and y_l.

Calculation of Jump Cost: The jump cost ($\mathfrak{C}$) for OSB is calculated by considering the set of all target sequences (B) to which a query sequence should be compared. In the first phase, the query sequence X is compared to a target sequence $Y \in B$, and we define:

$$C(X,Y) = \underset{i}{\text{mean}}(\min_{j}(\mathfrak{D}(x_i, y_j))) + \underset{i}{\text{std}}(\min_{j}(\mathfrak{D}(x_i, y_j))) \tag{3.31}$$

Thus, for every element x_i, we find the closest element y_j, and then we take the mean plus one standard deviation (std) of the distances to the closest elements. In order to ensure a fair comparison of sequence X to all target sequences $Y \in B$, the jump cost should be fixed. This is obtained by simply taking the mean of all the jump costs: $C(X) = mean\{C(X,Y)\}; Y \in B$

3.7.5 Continuous Dynamic Programming (CDP)

CDP [Oka, 1998] is able to perform subsequence matching (finding full query in longer target sequence) and to locate multiple occurrences of the query in the target series. For detail on CDP[3], please see [Oka, 1998]. The path cost matrix ($\mathfrak{P}$) of CDP is obtained in the following way:

$$\mathfrak{P}(j,i)\Big|_{i=1} = 3 \times \mathfrak{D}(j,1) \tag{3.32a}$$

$$\mathfrak{P}(j,i)\Big|_{i=2} = \min \begin{pmatrix} \mathfrak{P}(j-2,1) + 2 \times \mathfrak{D}(j-1,2) + \mathfrak{D}(j,2) \\ \mathfrak{P}(j-1,1) + 3 \times \mathfrak{D}(j,2) \\ \mathfrak{P}(j,1) + 3 \times \mathfrak{D}(j,2) \end{pmatrix} \tag{3.32b}$$

$$\mathfrak{P}(j,i)\Big|_{3\leq i\leq p} = \min \begin{pmatrix} \mathfrak{P}(j-2,i-1) + 2 \times \mathfrak{D}(j-1,i) + \mathfrak{D}(j,i) \\ \mathfrak{P}(j-1,i-1) + 3 \times \mathfrak{D}(j,i) \\ \mathfrak{P}(j-1,i-2) + \mathfrak{D}(j,i-1) + \mathfrak{D}(j,i) \end{pmatrix} \tag{3.32c}$$

The output is obtained by $A(j) = \frac{1}{3.p}\mathfrak{P}(j,p)$. where $\mathfrak{P}(j,p)$ is given by:

$$\mathfrak{P}(j,p) = \min_{(1\leq i\leq p,\, j\geq\beta(i),\, \beta(i+1)\geq\beta(i)} \sum_{i=1}^{i=p} \mathfrak{D}(x(i), y(j-\beta(i))$$

The above defined formula has this specific initial condition: $\mathfrak{P}(-1,\tau) = \mathfrak{P}(0,\tau) = \infty$. As

[3]The implementation of CDP, used here is taken from: http://www.diva-portal.org/smash/get/diva2:347724/FULLTEXT01.pdf. Page no. 86

it can be visible from Eq^n 3.32, more resistance is present for the diagonal link of the DP path compared to other two links [4].

3.7.6 Experimental Protocol and Results

To evaluate the performance of these sequence matching techniques, mentioned in this section, we performed following experiments. The results on segmented words from Dataset-1 and Dataset-2 are shown in Table 3.4[5] and Fig. 3.4. It is visible form the P-R curves and the mAP values shown in Fig. 3.13 and Table 3.4 that LCSS family has significant low accuracy compared to other matching algorithms. On Dataset-1, CDP has outperformed all other techniques in both the datasets, whereas OSB has also performed comparatively well. OSB is more stronger than LCSS for word image matching problems. Most probable reason is the architecture of OSB, which is capable of skipping outliers from query and target sequences. Along with it's noise skipping property, the added *jumpCost* for skipping some elements also plays a vital role in the performance of OSB. But high computational complexity of OSB due to it's inherent architecture, is a bottleneck for exploring it on bigger dataset such as Dataset-5 and Dataset-6.

Method Name	Accuracy (mAP)
LCSS	0.0354
1DLCSS	0.0489
DDLCSS	0.0489
OSB	0.2785
MVM	0.2026
CDP	0.3619

Table 3.4: Performance comparison of other relevant techniques on GW-15 dataset

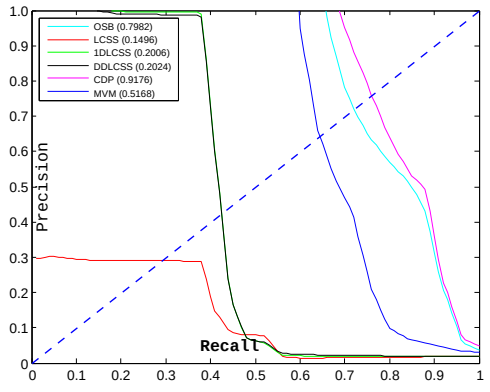


Figure 3.13: Performance comparison of other relevant techniques on Dataset-2

From the above explained experiments on Dataset-1 and Dataset-2, we see that the

[4]The above defined formula has the specific initial condition of $\mathfrak{P}(-1,\tau) = \mathfrak{P}(0,\tau) = \infty$. For implementation, two more rows are added at the beginning of distance matrix $\mathfrak{P}$, so the size of distance matrix become $(p+2) \times q$. The initial two rows of $\mathfrak{P}$ is filled up with inf and the above mentioned DP path formation starts from $\tau = 3$ which corresponds to the $1^{st}; (\tau = 1)$ row of the distance matrix $\mathfrak{D}$. At the time of locating the indexes at target sequence, where the query sequence has maximally corresponded with, we subtract the index values by 2, as initially the size of $\mathfrak{P}$ matrix was increased by 2. There are another way to consider the particular initial condition. In this case, the size of $\mathfrak{P}$ remains same and the initial two rows of this matrix are initialized with inf. It is seen that for our application, where the length of target and query sequence is quite large and always greater than 2, these aforementioned two different ways gives identical results.

[5]Please note that, the P-R plot of Dataset-1 is not shown because the accuracy of LCSS family (LCSS, 1DLCS, 2DLCSS) are very low compared to others. Due to this reason, when all the curves are put together in one graph, the curve of LCSS family is difficult to visualize. So, we prefer just to provide the statistical results.

performance of LCSS family is significantly lower than others. So, for the case of Dataset-3 and Dataset-4, we only tested other techniques except the LCSS family. The P-R plots are shown in the following Fig. 3.14. Thanks to the special DP-path and associated weights with these DP-Paths, CDP has outperformed MVM and OSB on both the datasets. That is why, instead of having the ability of skipping noise, FSM and OSB does not show better results than CDP. Whereas, thanks to extraction of slit style HOG based features from these two datasets, local level variation and noise effects could be more controlled. One of the main drawback of FSM and OSB is their inability to properly distinguish between inlier and outliers elements, hence to decide whether to skip or not. Due to this problem, FSM and OSB often takes wrong jumps of the inlier elements, whereas outliers are matched with query signal.

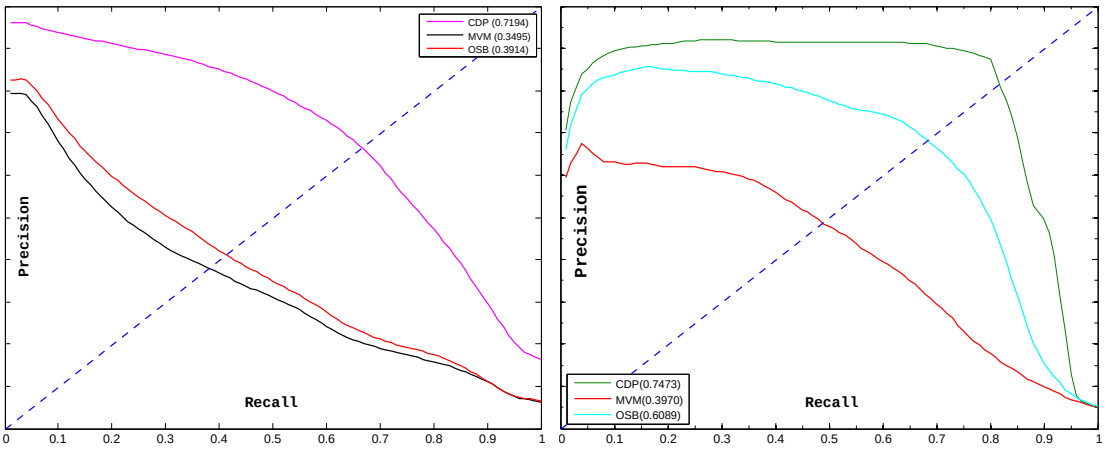
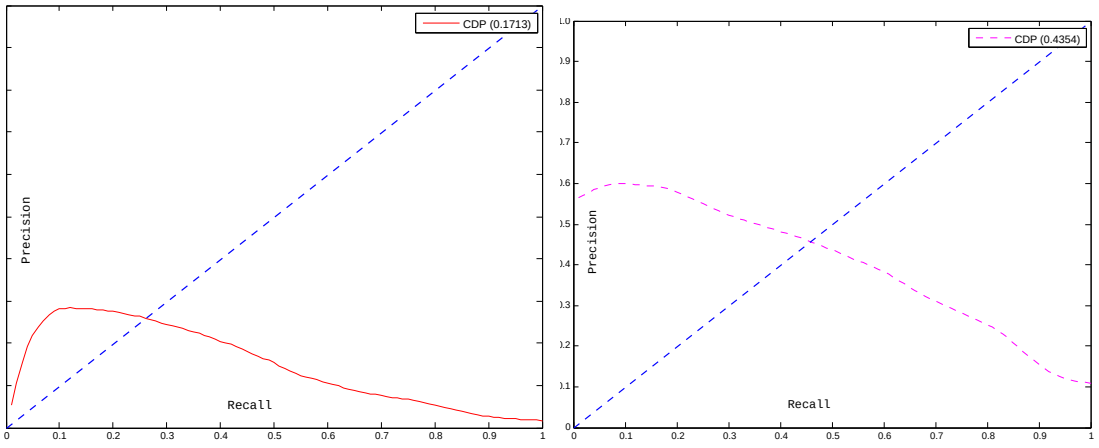


Figure 3.14: a) Performance comparison of other relevant sequence matching techniques on Dataset-3. b) Performance comparison of other relevant sequence matching techniques on Dataset-4.

In the above section, we observe that CDP has always performed better than all other techniques in all of the 4 datasets, whereas OSB has also performed comparatively well than others. But due to the high computational complexity of OSB, in this section, we only performed the experiments with CDP algorithm. The corresponding P-R curves and mAP values are shown in Fig. 3.15. The special DP-Paths and associated weights helps CDP to perform well on properly segmented word images.

For better visibility of mAP values of all the above mentioned algorithms, these values are mentioned in the following Table 3.5. The interesting performances by some specific algorithm is highlighted. From Table. 3.5, it can be visible that depending on the nature of the specific algorithms, most of these highlighted algorithms have shown credible performances on multiple datasets.

In Table 3.6, time needed to do the matching for one query of GW (to be matched over all words in the 10 pages) are reported. The experiments were performed, by using Intel i7 processor and 4 GB RAM. MatLab-7.14 is used for implementing algorithms except



(a) (b)

Figure 3.15: a) Performance comparison of different DP paths on Dataset-5. b) Performance comparison of different DP paths on Dataset-6.

Fast-DTW, in Java. For proper comparison, DTW was also implemented in Java. The ample difference in time factor between the two implementations is properly visible. As expected, we can see also that SC and Itekura Bands, as well as PDTW, are speeding up DTW. Finally, OSB and DTW-CW are much slower, but CDP is faster. Please note that intentionally, we have not shown the computation time of the remaining techniques as those are not significant in current perspective.

3.8 Combination of aforementioned sequence matching techniques

In this section, we present the approaches to combine different dynamic programming based sequence matching techniques for word spotting. This kinds of approaches of parametrically combining two different technique have been studied by other researchers, in the domain of time series matching [Górecki and Luczak, 2014] [Górecki and Luczak, 2015], but have not been much explored in the domain of document image processing. After observing the performance of different algorithms in the aforementioned section, we choose the best performing algorithms to parametrically combining them for having improvement in the result.

3.8.1 Constrained LDTW

We propose here to apply *Pseudo Local DTW*, constrained by *Itakura parallelogram*, for taking advantages of both the techniques. This fusioning helps to reduce the time and space complexity of LDTW and to avoid pathological matching similarly as for DTW. In the experimental section, we will show that this simple fusioning of Itakura parallelogram with LDTW improves the result to a good extend (3 to 4%). The corresponding pseudo

Table 3.5: Comparative word spotting accuracy of all the above mentioned sequence matching techniques on six datasets.

Technique	Dataset-1	Dataset-2	Dataset-3	Dataset-4	Dataset-5	Dataset-6
0-Symmetric	0.3668	0.6449	×	×	0.1323	0.3286
0-Asymmetric (classical DTW)	0.4576	0.8503	×	×	0.1573	0.4525
1-Symmetric	0.3074	0.6922	×	×	0.1180	0.3533
1-Asymmetric	0.2188	0.220	×	×	0.0166	0.0458
2-Symmetric	0.2038	0.5512	×	×	0.0513	0.1788
2-Asymmetric	0.1936	0.2029	×	×	0.0152	0.0417
3-Symmetric	0.4324	0.8515	×	×	0.1702	0.4389
0.5-Symmetric	0.2776	0.7168	×	×	0.1156	0.3901
0.5-Asymmetric	0.4642	0.8402	×	×	0.1663	0.4224
SC-Band	0.5876	0.7961	×	×	×	×
Itakura Parallelogram	0.6017	0.8013	×	×	0.2226	0.5265
DDTW	0.2180	0.5645	×	×	×	0.065
Value Derivative DTW	0.2265	0.6510	×	×	×	×
Weighted Hybrid DTW	0.3554	0.8621	×	×	0.146	0.358
PDTW	0.3466	0.8929	×	×	0.157	0.432
PDDTW	0.2563	0.9564	×	×	×	0.178
WDTW	0.3309	0.5854	×	×	×	×
WDDTW	0.2431	0.2319	×	×	×	×
LDTW	0.5374	0.7577	×	×	0.229	0.499
Isometric Sin DTW	0.4213	0.6790	×	×	0.102	0.335
Isometric Cos DTW	0.2426	0.5653	×	×	×	×
Isometric Hilbert DTW	0.5380	0.8907	×	×	0.245	0.566
SSDTW	0.3349	0.8409	0.645	0.7295	0.1181	0.3276
DTW-CW	0.3603	0.8756	0.643	0.734	0.1358	0.3537
MJ-DTW	0.3413	0.7779	×	×	0.1835	0.4370
LCSS	0.0354	0.1496	×	×	×	×
1DLCSS	0.0489	0.2006	×	×	×	×
DDLCSS	0.0489	0.2024	×	×	×	
OSB	0.2785	0.7982	0.3914	0.6089	×	×
MVM	0.2026	0.5168	0.3495	0.3970	×	×
CDP	0.3619	0.9176	0.7194	0.7473	0.1713	0.4354
Fast-DTW	0.4348	0.7806	×	×	×	×

code is given in Algorithm 3.

Algorithm 3: LDTW + Itakura Paralleogram

Input: $p, q, \mathfrak{D}$
Output: $\mathscr{D}$

```
for i ← 1 to p do
    for j ← 1 to q do
        bool = ITAKURA(i, j, p, q);
        if bool then
            Ω(i, j) ← LDTW(i, j, D);
D = D(p, q)/|w_k|                          ▷ The final distance
return;
```

Table 3.6: Time required for finding one keyword in GW.

Sr. No.	Algorithm Name	Time taken (Sec.)		Algorithm Name
1.	Classical DTW	602.85	397.24	SSDTW
2.	SC-Band	170.08	574.33	DDTW
3.	Itakura Band	240.00	155.28	PDTW
4.	0.5-Symmetric	672.98	172.44	PDDTW
5.	0.5-Asymmetric	575.95	1755.90	DTW-CW
6.	0-Symmetric	608.55	810.50	WDTW
7.	0-Asymmetric	567.62	825.43	WDDTW
8.	1-Symmetric	618.24	649.19	MVM
9.	1-Asymmetric	602.42	8474.15	OSB
10.	2-Symmetric	611.033	404.23	MJ-DTW
11.	2-Asymmetric	597.47	245.30	CDP
12.	3-Symmetric	519.43	208.54	LCSS
13.	LDTW	519.43	2.62	Fast-DTW
			5.23	DTW

3.8.2 Parametric Combination of Matching Techniques

We propose a simple approach to parametrically combine different dynamic programming based sequence matching techniques for word spotting. This was introduced for general time series signal matching domain [Górecki and Luczak, 2015] but we adapted it for word spotting purpose. In this section, the aforementioned matching algorithms are pairwise combined for obtaining characteristic advantages from both techniques.

3.8.2.1 Parametric Distance Measure

Let's consider two distance measures $\mathfrak{D}_1$ and $\mathfrak{D}_2$. A weighting based distance measure ($\mathfrak{D}_{ab}$) can be calculated as a linear combination of $\mathfrak{D}_1$ and $\mathfrak{D}_2$, based on two real parameters $a, b \in [0, 1]$ [Górecki and Luczak, 2015].

$$\mathfrak{D}_{ab}(X, Y) = a \times \mathfrak{D}_1(X, Y) + b \times \mathfrak{D}_2(X, Y) \tag{3.33}$$

It is noteworthy to mention that it is not needed to test all values of $a, b \in [0, 1]$. If $a_1 = ca_2$ and $b_1 = cb_2$, where $c > 0$ is a constant (i.e. the points (a_1, b_1) and (a_2, b_2) are linearly dependent), then the following Eq^n 3.34 is maintained by the selected parameters.

$$\mathfrak{D}_{a_1b_1}(X_1, Y_1) \lessgtr \mathfrak{D}_{a_1b_1}(X_2, Y_2) \Leftrightarrow \mathfrak{D}_{a_2b_2}(X_1, Y_1) \lessgtr \mathfrak{D}_{a_2b_2}(X_2, Y_2) \tag{3.34}$$

Due to this property of linear dependency, the parameters (a, b) can be chosen from any continuous line between points $(0, 1)$ and $(1, 0)$. Let's consider the case of straight line, given by:

$$a = (1 - \alpha); b = \alpha; \alpha \in [0, 1] \tag{3.35}$$

If the subset of parameter α is dense enough, the choice of parameterization should not be

critical. It is shown in the following section that any two methods can be parametrically combined by the above mentioned approach. For example, the parametric combination of DTW and DDTW is named as *parametric derivative DTW*, which was proposed by Gorecki et.al. in [Górecki and Luczak, 2015].

3.8.2.2 α–optimization technique

We also propose a way to obtain the weighting parameter from training samples (α). This parameter α should be tuned during training phase, by using only small training dataset. The requirement is to choose the best value of α, which gives highest mAP, on the learning dataset. Please note that the technique of α optimization is inspired by the technique mentioned in [Górecki and Luczak, 2015], but it is adapted according to our requirements for word spotting. This optimization can be applied on various other problems in the domain of document image processing and information retrieval.

To obtain the optimized value of α, first we select one query image $\mathfrak{Q}_1$ from the training dataset and we compute all distance values between this query and images from the target image set (T^{s^1}). Now, for every parameter of $\alpha(\{\alpha \in [0,1]\})$, we compute the distance function $d^1_{\{T^{1\ldots s^1}\},\{1\ldots p\}} = \mathfrak{D}_{ab}(\mathfrak{Q}_1, T^{1\ldots s^1})$ for $\alpha_{1,\ldots,p}$ and put it's value into a 2D matrix $\mathfrak{D}$ (with s^1 rows and p columns). Now we sort (descending order) each column separately of the $\mathfrak{D}$ matrix, as each column here represents the matches for different α values. As we have already the ground truth for this query image $\mathfrak{Q}_1$, we calculate the precision and recall (PR) values at each rank for every columns. Please note that, we need to compute the distance measures $\mathfrak{D}_1(\mathfrak{Q}_1, T^{1\ldots s^1})$ and $\mathfrak{D}_2(\mathfrak{Q}_1, T^{1\ldots s^1})$ only once for all the values of $\alpha_t; 0 \leq t \leq p$. Then for the next query image $\mathfrak{Q}_2 \neq \mathfrak{Q}_1$ from the training set, we repeat the procedure and obtain a new distance vector $d^2_{\{T^{1\ldots s^2}\},\{1\ldots p\}} = \mathfrak{D}_{ab}(\mathfrak{Q}_2, T^{1\ldots s^2})$. Then the same process of calculation of PR values is followed for each column of $d^2_{\{T^{1\ldots s^2}\},\{1\ldots p\}}$ matrix. This process is repeated for all the query images in the learning set i.e. until $d^q_{\{T^{1\ldots s^q}\},\{1\ldots p\}}$, where q represents the total number of query images present in the learning set. Now the idea is to find the best $\alpha_{1,\ldots,p}$, among all the query images. So, we calculate the mean average precision for each column (each $\alpha_{1,\ldots,p}$) for all the query images. Hence, we can get the best performing α on this learning set. After obtaining the optimized value of α, this α value is applied on the test set to compute the accuracy of the system.

For a given value of α, the complexity of computation of distances between the n elements $E_1, E_2, ..., E_n$ is $O(n^2)$. As the calculations of the distance functions $\mathfrak{D}_1$ and $\mathfrak{D}_2$ are the most time consuming part of optimization techniques, the computation time depends only to a small degree on the number of considered values for α (specially for large value of n). Due to this reason, it is possible to choose a large set of α values in the optimization process without increasing the computation time of the parameter tuning phase. If the minimum error rate is same for more than one value of α, the smallest α is chosen. Here the set of α values is chosen from 0 to 1 with fixed step 0.01.

3.8.3 Results

The accuracy of the proposed word spotting system is calculated in terms of mean average precision (mAP)[6] metric. In the following Table 3.7, we remind readers the mAP of best performing algorithms, which were mentioned before. It can be visible that, constraining classical DTW by *Itakura Parallelogram* (DTW) shows significant improvement in accuracy over DTW. *Pseudo Local DTW*, also shows significant improvement in accuracy over classical DTW, as previously observed. From the experimental results, we can finally observe that the proposed fusioning of LDTW with Itakura parallelogram (LDTW) also shows interesting performance. The mAP of the system for Bentham dataset is **0.524** and for GW dataset it is **0.276**. So, no significant improvement is obtained for Bentham dataset, compared to the performance of classical Itakura parallelogram but nice improvement is achieved for GW dataset ($+4, +7$ %). One possible reason for this is that, inside the band of Itakura parallelogram, LDTW's different DP paths and associated weights are not able to add extra values for the Bentham dataset because of less presence of noise compared to GW dataset. Moreover, it can also be observed from the experiments that the concept of piecewise aggregate approximations of signals (PDTW) does not show better performance than classical DTW. Most probably the aggregation of local information by agglomerating the features, extracted from single pixel width columns in not a good idea for word spotting in hand written, historical dataset. Most probably, capturing the local level variations in word image signals are important for word image matching. At last but not least, SSDTW as well as CDP does not show better performance than DTW because they are specially designed to find subsequences. Since we use well segmented word images for the experiment, in such cases, classical DTW is more suitable choice than SSDTW or CDP.

Table 3.7: mAP of different algorithms, applied on Bentham (1^{st} row) and GW (2^{nd} row) dataset.

DTW	**PDTW**	**CDP**	**SSDTW**	**DTW**	**LDTW**	**LDTW**
0.45	0.43	0.44	0.33	**0.52**	**0.499**	**0.524**
0.178	0.157	0.171	0.118	**0.229**	**0.229**	**0.276**

After seeing the results of individual sequence matching techniques, in the following section, we performed experiments with parametric combination of these individual techniques. The rows and columns of Table 3.8 and Table 3.9, represents the result of parametric combination of different algorithms. The matching techniques, mentioned row wise in Table 3.8 and Table 3.9 represents, $\mathfrak{D}_1$ and the columns represents, $\mathfrak{D}_2$ (see Eq^n 3.33). To evaluate the combined approaches independently to the choice of the learning set, learning and testing steps are repeated 10 times. Each time 2 queries are randomly selected as learning set, for getting optimized value of α and then testing is performed on the remaining query images. The Table 2 and 3 shows the average mAP obtained in this manner.

As visible from Table 3.8, the parametric combination of CDP & LDTW (0.279) and DTW & LDTW (0.280) has performed well for the case of GW dataset. Similar kinds of outcome are also visible for Bentham dataset. The parametric combination of CDP

[6] `http://en.wikipedia.org/wiki/Information_retrieval\#Mean_average_precision`

Table 3.8: mAP of parameterized matching techniques for GW dataset

	DTW	CDP	LDTW	SSDTW	PDTW	LDTW	Itekura
DTW		0.214	0.232	0.185	0.181	0.262	0.221
CDP	0.205		0.237	0.162	0.194	**0.279**	0.238
LDTW	0.239	0.244		0.229	0.234	0.277	0.254
SSDTW	0.188	0.168	0.223		0.168	0.264	0.211
PDTW	0.186	0.188	0.227	0.171		0.265	0.214
LDTW	0.273	0.271	0.272	0.270	0.264		0.272
Itekura	0.220	0.236	0.250	0.214	0.217	**0.280**	

Table 3.9: mAP of parameterized matching techniques for Bentham dataset

	DTW	CDP	LDTW	SSDTW	PDTW	LDTW	Itekura
DTW		0.468	0.496	0.436	0.433	0.529	0.503
CDP	0.461		0.504	0.412	0.443	**0.540**	0.520
LDTW	0.492	0.495		0.467	0.501	0.515	0.523
SSDTW	0.410	0.410	0.477		0.416	0.508	0.511
PDTW	0.434	0.462	0.489	0.422		0.533	0.506
LDTW	**0.543**	0.519	0.514	0.518	0.552		0.536
Itekura	0.503	0.529	0.522	0.511	0.510	**0.539**	

LDTW = LDTW+Itekura

& LDTW (0.540), DTW & LDTW (0.539) and LDTW & DTW (0.543) has performed well. **Please note that the accuracy (best until now, as far our knowledge is concerned) of proposed technique has considerably outperformed the base line accuracy of 0.407 for Bentham dataset, for segmentation based approaches**[7]. It can be observed that our system, based on classical DTW has also outperformed the base-line system. Most probably, it is due to our pruning technique and selected column based features [Mondal et al., 2014].

Table 3.10: Comparative word spotting accuracy of GW dataset.

Technique	**mAP**
Proposed	**0.28**
[Wang et al., 2014b]	0.175
BOVW (see [Wang et al., 2014b])	**0.422**
Pseudo-Struct (see [Wang et al., 2014b])	0.072
Structural (see [Wang et al., 2014b])	0.028

The accuracy of our system for GW dataset could be compared with the one in [Wang et al., 2014b]. Although the used queries in [Wang et al., 2014b] are different than the queries used in this chapter, the mAP obtained by DTW based system (see Table 1 in [Wang et al., 2014b]) is almost equal to our DTW based system's accuracy (0.178), shown in Table 3.7. Hence, it can be asserted that our chosen queries are analogous to the queries chosen in [Wang et al., 2014b]. Based on this conjecture, it is apparent that **our system's accuracy is the second best among the notable state-of-the-art word spotting methods, mentioned in [Wang et al., 2014b]** (see Table 3.10).

[7]http://transcriptorium.eu/ icdar15kws/evaluation.html. Please note that the experiment was performed only on validation set (not on test set). The ICDAR-2015 competition result is published only on test set and the best reported result is 0.424 (mAP).

3.9 Conclusion

In this chapter, different dynamic programming based matching techniques are explored for word spotting. It is visible from the experiments that constraint based DTW and DTW with different DP paths and weights can outperform classical DTW. It is visible from the results on segmented words that piecewise aggregation of image features and transforming the image features by several above mentioned transforms (e.g. Hilbert, Sin etc.) can significantly improve the results. Weight based hybridization of original signals and their derivatives are more effective for word spotting than taking only the derivative of the signals. Moreover, it is also demonstrated in the chapter that parameterized combination of various DP path based matching techniques can highly improve the accuracy. In future, we would like to explore other dynamic programming based approaches for word spotting.

Chapter 4

Flexible Sequence Matching

Contents

Abstract

In the previous chapter, we have investigated several sequence matching techniques for word spotting. We have also discussed the pros and cons of these techniques. In this chapter, a robust method is presented to perform word spotting in degraded hand written and printed document images. A new sequence matching technique, called as Flexible Sequence Matching (FSM) algorithm is introduced for this task of word spotting by analyzing the drawbacks and advantages of other sequence matching techniques. The FSM algorithm was specially designed by incorporating beneficial characteristics of other sequence matching algorithms especially Dynamic Time Warping (DTW), Sub-sequence DTW (SSDTW), Minimal Variance Matching (MVM) and Continuous Dynamic Programming (CDP). Along with the characteristics of multiple matching (many-to-one and one-to-many), FSM is also capable of skipping existing outliers or noisy elements, irrespective of it's position in the target signal. More precisely, in the domain of word spotting, FSM has the ability to retrieve complete words or words containing only a part of the query. Furthermore, due to it's adaptable skipping capability, FSM is also less sensible to local variations in the spelling of words, and also to local degradation effects within the word image. The multiple match-

ing capability (many-to-one, one-to-many) of FSM helps it to deal with stretching effects of query and/or target images. Moreover, FSM is designed in such a manner that with little modifications, it's architecture can be changed into the architecture of DTW, MVM, SSDTW, and CDP. To illustrate these possibilities of FSM, we performed experiments on incorrectly segmented words and on words with local spelling variations. We have also considered properly segmented lines of historical handwritten documents of different languages and improperly as well as properly segmented words of type written and handwritten historical documents. From the comparative experimental results of word spotting, it can be clearly visible that the proposed FSM technique outperform other sequence matching approaches except CDP. It is shown in the experimental section that although FSM could not outperform CDP in overall statistical results (mAP) but it shows better individual statistical results (mAP) than CDP for some certain queries.

4.1 Highlights on the Properties of Main Sequence Matching Techniques

As explained in Chapter 3, numerous sequence matching methods have been proposed till date. The most popular and most used one is Dynamic time warping (DTW) [Albrecht, 2009], [Itakura, 1975], [Niennattrakul and Ratanamahatana, 2007], [Yu et al., 2007], which has shown promising results for time series classification and clustering because of it's non linear mapping capabilities. It has been widely used for comparing time series data for data classification, clustering and word spotting. The main idea of DTW is to calculate the distance between the time series by summing up the distances of their corresponding elements. DTW yields an optimal (order preserving) relation $\mathbb{R}$ of all elements of sequence $x = \{x_1, x_2, x_3....x_p\}$ to all elements of sequence $y = \{y_1, y_2, y_3....y_q\}$. Dynamic programming is used to find the best corresponding elements. It has been shown in literature that DTW distance is superior to Euclidean distance [Vlachos et al., 2002] for many applications including word spotting [R. Manmatha, 2003]. Nevertheless some limitations exist with classical DTW. Especially each element of $x_{1..p}$ must corresponds to some $y_{1...q}$ and vice versa (one-to-one, one-to-many or many-to-one matchings). This hard constraint often forces DTW to perform correspondence with noisy elements, which could disturb the matching and could also increase final distance value, especially in the applications, where high amount of noise is present in signal, which particularly true for word spotting. Hence, ranking of compared elements can be disturbed. Another problem arises when sequence x corresponds to only a part $y^{'}$ of the sequence y, DTW can not ignore the elements that do not belong to $y^{'}$. To perform partial matching of sequences, some research have been introduced in the literature e.g. SSDTW [Albrecht, 2009], which is designed to find a continuous subsequence within a longer sequence that can optimally fit the shorter query sequence. Another DTW based modification, called as *"DTW with corresponding window" (DTW-CW)* [Latecki et al., 2007b], is able to perform partial sequence matching, using a sliding correspondence window of same size as the query sequence. Although this approach can give high comparable accuracy, it is also highly time consuming, due to it's architectural structure. It also has similar drawbacks as DTW, especially the inability to skip noise. Moreover, deciding the threshold for sliding the corresponding window is highly

data dependent and a cumbersome process.

There are several variants of DTW exists in literature. Among these ones, some have different architectural structure, such as Longest Common Subsequence (LCSS) [Vlachos et al., 2002], Minimal Variance Matching (MVM) [Latecki et al., 2007b], Optimal Sequence Bijection (OSB) [Latecki et al., 2007a],

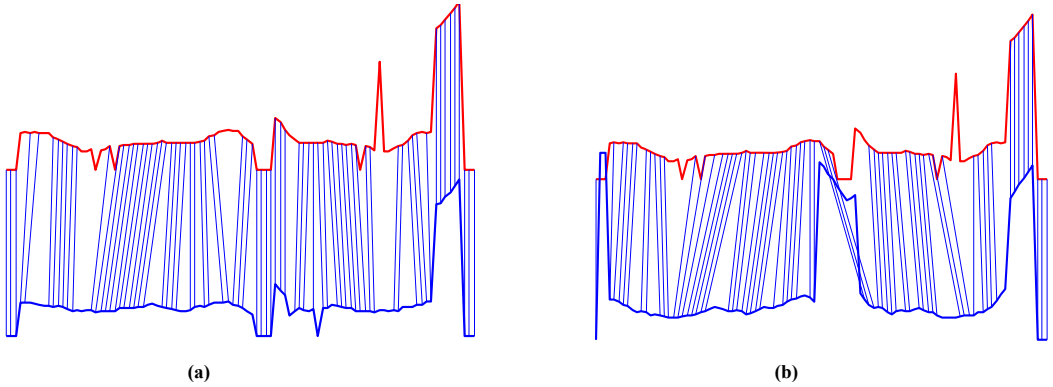


Figure 4.1: In LCSS, there is no penalty for skipping noisy elements, which often generates wrong correspondences. (a) The query sequence (top one) is similar to the target sequence (bottom one) (b) the target sequence is different than the same query sequence. Although the length of the optimal subsequences is 73 in both cases[1].

To find an optimal correspondence between two sequences, a popular approach is Longest Common Subsequence (LCSS) [Vlachos et al., 2002] [Vlachos et al., 2003]. Given a query and a target sequence, LCSS determines their longest common subsequence i.e. the sequence that best correspond with each other, allowing skipping from both sequences (see Fig. 4.1). The dissimilarity measure is based on the ratio between length of longest common sub-sequence and the length of the whole sequence. The elements of the subsequence do not need to be consecutive points, which could be a problem for word spotting. Order of points is not rearranged and some points can remain unmatched. But to make LCSS efficient, one needs to set a threshold that determines when values of corresponding points would be treated as equal or not. There are no automatic approach to determine this threshold value and it is always has to be settled manually [Das et al., 1997]. The performance of LCSS is highly dependent and sensitive on correct setting of this threshold value (see Fig. 4.1). Compared to euclidean distance, DTW and LCSS are more elastic, supporting local time shifts and variations in the lengths of pairs of time series, but they are also more expensive to compute.

[1]Figure is used in this thesis, with written permission from [Longin Jan Latecki, Suzan Koknar-tezel, Qiang Wang, Megalooikonomou Vasileios, "Sequence Matching Capable of Excluding Outliers", In Proceedings of Workshop on Time Series Classification at ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD), 2007.]

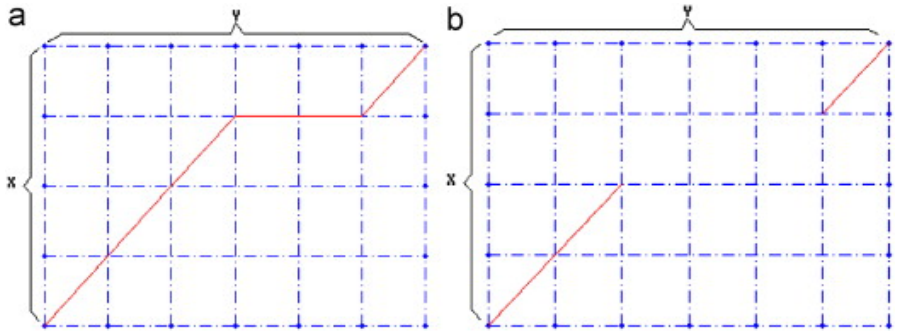


Figure 4.2: Alignment of query and target time series by a) DTW b) MVM[2].

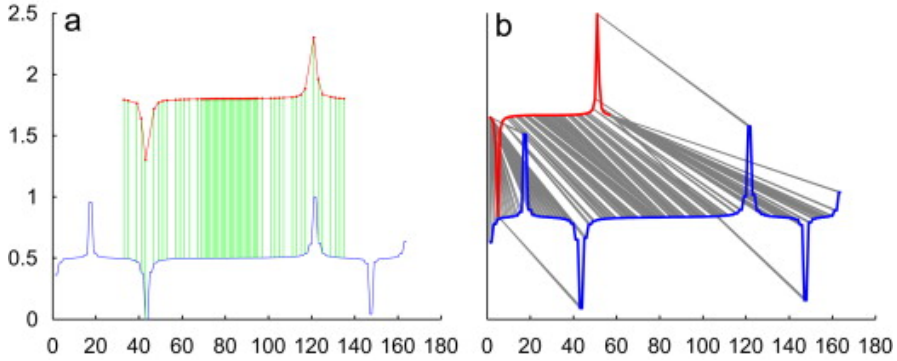


Figure 4.3: (a) MVM is able to generate the correct correspondence for a complete query matched with part of target sequence. (b) DTW is not able to create correct correspondence since it can not skip outliers elements[2].

To overcome these restrictions of LCSS, Minimal Variance Matching (MVM) proposed by Lateki et. al. [Latecki et al., 2007b]; by combining the strength of both DTW and LCSS, while overcoming their constraints. MVM calculates the sequence similarity directly based on the distances of corresponding elements (see Fig. 4.3). MVM also tries to find an optimal path including all the corresponding pairs but MVM is able to skip outliers in target during the matching process, hence it can handle partial matching and the calculated optimal path including all the corresponding pairs does not need to be consecutive. The notable difference between LCSS and MVM is that LCSS optimizes the length of the longest common sub-sequence (based on the distance threshold), while MVM optimizes the sum of distances of corresponding elements (without any distance threshold). LCSS is able to skip both query and target elements whereas MVM can only skip target sequence. So, MVM could be used only when the query sequence is smaller than target sequence and it is useful to find the query in a bigger target sequence (see Fig. 4.3). For the case of word spotting, this can be ideal condition, if we consider that the query image is perfectly selected and there is no possible noise present inside. But when there are outliers in query sequence

[2]Figure is used in this thesis, with written permission from [Longin Jan Latecki, Vasileios Megalooikonomou, Qiang Wang, Deguang Yu, "An elastic partial shape matching technique", Pattern Recognition, 40(11), 2007. © 2007 Elsevier.]

and skipping these ones are necessary then MVM is not a good choice compared to LCSS. Also both algorithms (LCSS and MVM) are only capable to do one to one correspondences and are unable to perform many to one and one to many matching. Moreover, there is no penalty for jumping some element in MVM, thus it does not create any resistance for jumping some element. This complete freedom to jump, sometimes generate non-relevant matching. MVM is also unable to differentiate between real noisy element and minor degraded elements which could leads to unintuitive correspondence of elements.

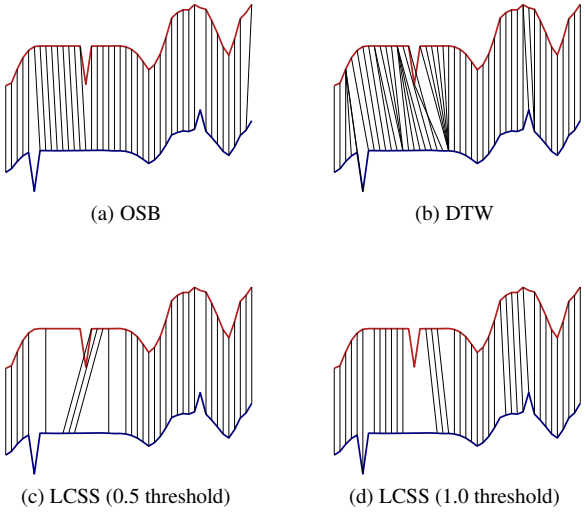


Figure 4.4: The query (top) and target (bottom) sequences looks very similar but are obtained from different sources. (a) The correspondence obtained by OSB; (b) by DTW, which is unable to give correct correspondence in the presence of outliers. (c)(d) The alignment by LCSS with different threshold are also not able to provide good correspondences[3].

To overcome these bottlenecks, another algorithm was designed by Lateki et. al., known as: *Optimal Subsequence Bijection (OSB)* [Latecki et al., 2007c], which can ignore outliers elements from query and target sequences. The distance between two sequences is also given by the sum of distances of their corresponding elements (see Fig.4.4). The goal of OSB is to find subsequence $x^{'}$ from x and a subsequence $y^{'}$ from y such that $x^{'}$ best matches $y^{'}$, skipping some elements from x and y as both may contain some outliers elements. So, by giving one to one correspondence, OSB is able to find the common sub-sequence between query and target series. Unlike LCSS, it does not require any distance threshold. OSB penalizes the distance for skipping consecutive elements proportionally to the number of elements skipped thus skipping one outlier costs less than skipping consecutive successive elements. Since there is no penalty for skipping elements in the case of LCSS, this makes it vulnerable to accidental matches. The main difference between OSB and MVM is that OSB can skip outliers from both the query and target sequences, whereas MVM can only skip noise from target sequence. Also, as already mentioned, there is no penalty for skipping in MVM. One crucial drawback of OSB is it's high computational complexity. Another

[3]Figure is used in this thesis, with written permission from [Longin Jan Latecki, Suzan Koknar-tezel, Qiang Wang, Megalooikonomou Vasileios, "Optimal Subsequence Bijection", ICDM, 2007.]

crucial drawback in MVM, OSB, and LCSS is that, none of them support one to many and many to one matching, contrary to DTW, which is comparatively robust in many applications especially when there are some stretch present in either or both query and target sequences. The following Tables. 4.1 and 4.2, summarizes the main properties of different algorithms.

Based on previous conclusions, this chapter (next section) proposes a novel technique which combines all the aforementioned facilities of MVM and DTW, along with some special features, taken from OSB. Indeed our proposed technique is capable of skipping outliers from target series, as MVM. So, it is able to find best corresponding elements from target sequence by avoiding noise and allowing partial matching. Like DTW, our proposed technique is also capable to perform multiple matching, which helps to handle the presence of stretching and/or contractions in the signal. In addition, a penalty for skipping elements of target sequence (as in OSB) is added to keep into account the amount of noise present. This helps the algorithm to properly indexing target sequences with respect to the query sequence.

4.2 Flexible Sequence Matching

The architecture of FSM[4] is designed in such a manner, so that it can have all the individual advantages of other sequence matching techniques into one. In this section, the complete mathematical structure of FSM is described. Please note that the main characteristics of FSM are presented in Table 4.1 and 4.2. FSM creates a relation $\mathbb{R}$ between two finite sequences x (query) and y (target), of different lengths[5] p and q: $x = (x_1, x_2, ..., x_p)$ and $y = (y_1, y_2, ..., y_q); p \leq q$. The goal of the algorithm is to find $y^{'}(y^{'} \subset y)$ such that x best matches with $y^{'}$ under given constraint. The relation $\mathbb{R}$, is performed on the set of indices $\{1, ...p\} \times \{1, ...q\}$, where, one-to-one, one-to-many and many-to-one mapping are possible. Based on the correspondence found, a distance between the two sequences is computed. A theoretical description as well as an algorithmic description are given in the following subsections.

4.2.1 Theoretical Description of FSM

The theoretical background of the FSM algorithm comes from both DTW and MVM algorithms. We follow the same relational structure of DTW (one to one, many to one and one to many matching) in order preserving manner, while the requirement for the participation of all the elements of y in the relation $\mathbb{R}$ could be relaxed. It means, if there are some noisy elements present in y then, they can be intelligently skipped by the algorithm. This proposed modification of the algorithm allow us to limit the influence of outliers in y, since x could find the best match with the sub-sequence $y^{'}$ of y. Note that,

[4]Full implementation of FSM is available at : `https://github.com/tanmayGIT/FSM`

[5]Please note that, for FSM, the length of query sequence should always be less than or equal to the length of target sequence. If the length of query sequence is longer than target sequence, then we can treat the original query sequence as target and original target sequence as query, i.e. we just reverse the order of input arguments in FSM function.

Table 4.1: Characteristics of different sequence matching techniques

$\mathcal{MN}$	$\mathcal{CS}$	$\mathcal{BC}$	$\mathcal{NC}$	$\mathcal{OS}$
DTW	Each query element has to be matched with target elements without skipping any elements from target and query sequence (and vice versa).	Correspondence involves every elements of the target and query sequence, including 1^{st} and last elements	One-One, Many-One, One-Many	Suitable for query and target sequence similar length.
SSDTW, CDP	Partial sequence matching is possible. Skipping outliers only at the end and beginning of the target signal is possible.	Every query elements are involved but the correspondence with target signal can start and end at any position of the target sequence.	One-One, Many-One, One-Many.	Unlike DTW and other algorithm, No element wise correspondences are provided for CDP.
LCSS	Skipping of query and target elements are possible (no skipping penalty).	Correspondence between target and query signal can start and end at any position of the query and target sequence.	One-One	Threshold to define when two elements are similar. Partial sequence matching is possible.
MVM	Each query element has to be matched with one target element but not vice versa. Skipping is possible at any position of target signal, i.e. at the beginning, end and inside matched sequence.	Every elements of query signal are involved but correspondences with target signal can start and end at any position.	One-One	Query sequence has to be smaller than target and there are no skipping penalty. Partial sequence matching is possible.
OSB	Skipping of elements at any position of query and target signals are possible (penalty is added for skipping).	Same as LCSS	One-One.	Skipping penalty. Partial sequence matching is possible.
FSM	Each query elements has to be matched with some target elements but not vice versa. Skipping is possible at any position of target signal.	Same as MVM.	One-One, Many-One, One-Many	Skipping penalty. Query length has to be less than equal to target length. Partial sequence matching is possible.

$\mathcal{MN}$= Methods Name; $\mathcal{CS}$ = Continuity (Skipping); $\mathcal{BC}$ = Boundary Condition
$\mathcal{NC}$ = Nature or correspondence; $\mathcal{OS}$ = Other specificities.

1D time series signals are considered here. However, the extension to sequences of elements corresponding to vectors of same dimension is straight forward.

First the difference matrix $\mathfrak{D}$ is calculated using Euclidean distance $\mathfrak{D}_{i,j}=\sqrt{(y_j - x_i)^2}$; $1 \leq i \leq p; 1 \leq j \leq q$. According to [Latecki et al., 2007b], there is no restriction on using

Table 4.2: Comparison of Time complexities of different sequence matching techniques

Method Name	**Time Complexity** (Where, p and q represents the sizes of sequences, to matched)
DTW [Albrecht, 2009]	The complexity of DTW is $O(pq)$. It can be reduced to $O(p)$ with Sakeo-Chiba [Albrecht, 2009] and Itekura bands [Albrecht, 2009].
SSDTW [Albrecht, 2009]	Same as DTW
LCSS [Vlachos et al., 2002]	Same as DTW
MVM [Latecki et al., 2007b]	The complexity of MVM is $O(pq^2)$. But if *"Corresponding window bound"* (refer to [Latecki et al., 2007b]) is introduced, the complexity can be reduced to $O(pq)$. Moreover, when the query sequence is to be matched with complete target sequence, the complexity is reduced to $O(p)$.
OSB [Latecki et al., 2007c]	The time complexity of OSB is $O(p^2q^2)$. By imposing warping window restriction on OSB, and by limiting the number of elements that can be jumped, we can reduce the time complexity of OSB up-to $O(p)$.
FSM [Proposed]	The time complexity of FSM is $\Theta(3pq^2)$ (see section 4.2.4).

various distance measures and any of the distance measure can be considered.

The obtained difference matrix[6] $\mathfrak{D}$ can be used to generate a directed acyclic graph (DAG) denoted as $\mathbb{G}$, where each of $\mathfrak{D}$'s elements can be considered as a node. The links between each node and it's child nodes is obtained by solving a shortest path problem in this difference matrix, i.e. by using the cost function $\mathcal{H}$ defined in Eq^n 4.2, where $\mathfrak{D}_{u,k}$ matrix represents parent nodes and $\mathfrak{D}_{i,j}$ represents child nodes.

$$\mathcal{L} = (k+1) + elasticity - |k-u|; elasticity = |q-p| \tag{4.1}$$

$$\mathcal{H}(\mathfrak{D}_{u,k}, \mathfrak{D}_{i,j}) = \begin{cases} \mathfrak{D}_{i,j} & (i) \quad \text{if } i = u+1 \text{ and } k \leq j \leq \mathcal{L}; \quad (ii) \quad \text{if } i = u \text{ and } j = k+1 \\ \infty & \text{otherwise} \end{cases} \tag{4.2}$$

In Eq^n 4.2, any node, $\mathfrak{D}_{u,k}$ has the flexibility to be connected with other node $\mathfrak{D}_{i,j}$ in two possible ways: ((i) and (ii)). The condition (i) says that, a parent node can be connected with child nodes at the next row ($i = u+1$), at the same column or at the right ($k \leq j \leq \mathcal{L}$), up to certain user defined elastic limit [7] $\mathcal{L}$, with respect to the position of parent node.

[6] Please note that the indexes of all the matrix notations used in this chapter, always start from 1.

[7] When $q = p$, the value of *elasticity* is taken as 2 to have the outliers skipping facility. If the value of

This elastic limit (Eq^n 4.1) represents the number of jumps that could be performed inside the matched part of the target. By default, it is equal to the difference in size, between the query and target sequence ($|q-p|$) minus the number of jumps already used, i.e. the difference between column and row index of parent node: $|k-u|$. This constraint is not mandatory but it limits the complexity of the algorithm. The connection at the same column ($k=1$) is mainly to ensure the many to one matching (link with the parent node and the child node at the same column). The condition (ii) says that the parent node can also be connected with the node just next to it, at the same row. This particular connection, ensures the one to many matching facility. By this manner, we can generate $\mathbb{G}$ (see Fig. 4.6).

Judging the intensity of noise, present in the signal is a difficult and puzzling problem. Indeed, deciding, whether a particular element should be considered as an outlier (hence skipping it) or not (hence matching it) is not straight forward. For example, in the case of LCSS, the user need to define a threshold for this purpose. In FSM, these kind of threshold is not required, but the system can not be allowed to skip outliers without any resistance. Otherwise, the process can skip frequently/easily the elements, instead of matching them with some acceptable dissimilarity cost. So, to limit skips, a *skipCost* is introduced for penalizing skips. This penalty plays a vital role in distance computation and thus on overall ranking of the closest matches in time series retrieval (e.g. word images retrieval). To compute it for a specific problem, we randomly choose two query signals and for each one, we randomly choose two target signals, which are similar to the query (e.g. for the case of word image retrieval, word images which has same ASCII transcription as query). After selecting query and target signal pairs, the distance matrix ($\mathfrak{D}$) between the feature vectors of query and targets are calculated. A vector $\mathfrak{M}_{\mathfrak{i}=1..\mathfrak{p}}$ is created to contain the average of top m minimum values from each rows of $\mathfrak{D}$. The obtained vector $\mathfrak{M}_{\mathfrak{i}}$ from each query and target pairs are merged together and sorted in ascending order ($\mathfrak{M}^{merged}$). To have a good approximation of *skipCost*, we choose b% of the total number elements from $\mathfrak{M}^{merged}$ and keep them in another vector called $\mathfrak{M}_{\mathfrak{b}}{}^{merged}$. After obtaining these b% (we took b = 90, for our experiments) of total elements [8], the *skipCost* is then obtained by calculating mean and standard deviation of $\mathfrak{M}_{\mathfrak{b}}{}^{merged}$ (see Eq^n 4.3). The idea of choosing m minimum values is to have a cost that will not be null (and also give good approximation of skip cost), when perfect matches are available between query and target sequences. It also gives a better approximation of *skipCost*[9].

$$\begin{aligned} \forall i_{i=1,\ldots,p}, \quad \mathfrak{M}_i &= mean\{\underset{j=1,\ldots,q}{\text{top-}m\_\min}(\mathfrak{D}_{i,j})\}; \\ \mathfrak{S} = skipCost &= mean(\mathfrak{M}_{\mathfrak{b}}{}^{merged}) + 2 \times std(\mathfrak{M}_{\mathfrak{b}}{}^{merged}) \end{aligned} \tag{4.3}$$

To perform the matching, the main objective is to find the shortest path from one initial node $\mathfrak{D}_{1,j}$, leading to $\mathfrak{D}_{p,l}; l \in \{p....q\}$, through $\mathbb{G}$. The path cost (denoted as $\mathfrak{P}(\mathfrak{i},\mathfrak{j}); 1 \leq$

elasticity becomes 0 then FSM would not be able to skip outliers and would behave similarly as DTW.

[8] Please note that if after considering the elements of a particular bin, the considered number of elements exceeds 90% of total elements, then it is not a problem

[9] $m = 2$ is the minimum to consider but experimentally, taking $m = 5$ is less sensible and in acceptable range, with respect to the size of our sequences. This parameter is not very important for the proposed algorithm

$i \leq p; 1 \leq j \leq q$) is obtained by satisfying the following:

i) Starting at the 1^{st} row between column 1 and q i.e. at $\mathfrak{D}_{1,j}$ for $j \in \{1...q\}$ (this allows to skip unnecessary prefix)[10].

ii) Ending at the last row $\mathfrak{D}_{p,l}$; for $l \in \{p....q\}$, (to skip unnecessary suffix).

iii) The shortest path between each pair of reachable nodes in $\mathbb{G}$ can be found from $\mathfrak{P}$.

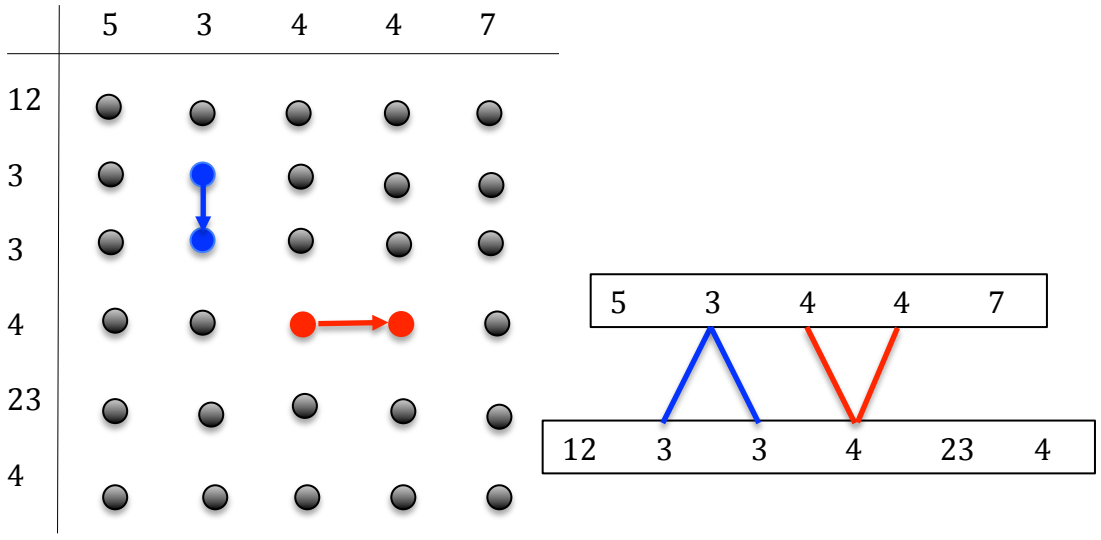
The following Eq^n 4.4a says that the 1^{st} row of the distance matrix $\mathfrak{D}$ is copied to the path cost matrix $\mathfrak{P}$. Each cell that belongs to the i^{th} row is calculated (Eq^n 4.4c) by first choosing the possible parent nodes at previous row $(i-1)$ and at the columns k ranging from $((i-1) - elasticity)$ to $((i-1) + elasticity)$. With respect to each of these parent nodes, the child nodes can only belongs (according to the construction of $\mathbb{G}$) to the next row and at columns ranging from the next column $(k+1)$ with respect to parent node until $(k+1) + elasticity - |k-(i-1)|^{th}$ column. Other possible links are for many-to-one and one-to-many matching: the link just below $(i-1, j)$ to (i, j) is responsible for one-to-many matching (see the blue link in Fig.4.5a) and the link just at left $(i, j-1)$ to (i, j) is responsible for many-to-one matching (see the red link in Fig.4.5a). Please note that in such cases, a small penalty ($\mathfrak{C} = mean(\mathfrak{M}^{merged})$) could be introduced to limit the numbers of many-to-one and one-to-many matching. It is noteworthy to mention that, in certain applications, many-to-one and one-to-many characteristics of FSM may not be required. In such cases, the last two terms of Eq^n 4.4b, could be ignored. The path cost $\mathfrak{P}(i, j)$ is updated only when there is a shorter path coming from a parent node. The optimal structure condition guarantees that the returned matrix $\mathfrak{P}$, contains the cost of the shortest path leading to every node. The total path cost between two comparable sequences can be obtained by getting the minimum value stored at the last row and l^{th} column of path cost matrix $\mathfrak{P}$, where $p \leq l \leq q$. To obtain the distance between two sequences, we normalize the dissimilarity value, through dividing it by the number of corresponding pairs between the target and query sequence (length of the warping path)[11].

$$\mathfrak{P}(1, j) = \mathfrak{D}_{1,j} \text{ if } 1 \leq j \leq q \tag{4.4a}$$

$$\mathfrak{P}(i, j) = \left(\min \begin{pmatrix} \{\mathfrak{P}(i-1, k) + \mathfrak{D}_{i,j} + (\mathfrak{S} \times (j - (k+1)))\} \\ \{\mathfrak{P}(i, j-1) + \mathfrak{C} + \mathfrak{D}_{i,j}\} \\ \{\mathfrak{P}(i-1, j) + \mathfrak{C} + \mathfrak{D}_{i,j}\} \end{pmatrix} \text{if } \mathfrak{L} \right) \tag{4.4b}$$

[10]Please note that to allow substring matching, these skips at the beginning and at the end are not penalized (no skipCost added)

[11]Some other normalization technique has also been adapted; those normalization techniques are described in the concerned experimental section



(a) (b)

Figure 4.5: (a) The illustration of one-to-many (blue) and many-to-one (red) matching on toy examples. (b) Corresponding matching elements of the query and target vectors (shown at the left).

$$\mathfrak{L} = \begin{pmatrix} 2 \leq i \leq p \\ max[1,(i-1)-elasticity] \leq k \leq min[q,(i-1)+elasticity] \\ k+1 \leq j \leq min(q,(k+1)+elasticity-max(0,\{k-(i-1)\})) \end{pmatrix} \quad (4.4c)$$

FSM creates a relation $\mathbb{R}$ from two finite sequence x (query) to y (target) of different lengths p and q: $x = (x_1, x_2, ..., x_p)$ and $y = (y_1, y_2, ..., y_q); p \leq q$. To align these two sequences using FSM algorithm, a difference matrix ($\mathfrak{D}$) of size $q \times p$ is constructed, where $\mathfrak{D}(i,j)$ is the distance between x_i and y_j. A warping path $\mathfrak{W}$ is a contiguous set of matrix elements that define a mapping between the series X and Y. The k^{th} element of $\mathfrak{W}$ is defined as $\mathfrak{W}_k = (i,j)_k$. The complete warping path is then defined as $\mathfrak{W} = \mathfrak{W}_1, \mathfrak{W}_2, \mathfrak{W}_3, ..., \mathfrak{W}_k, ...\mathfrak{W}_K; p \leq K \leq q + p - 1$. The upper bound of maximum number of elements possible (q+p-1) in warping path ($\mathfrak{W}$) of FSM is rationalized from the mathematical rationale of DTW, which also maintains the same upper limit.

Generally the warping path also follow some constraint:

i. **Boundary Condition** : FSM does not maintain the boundary condition of classical DTW and able to do partial sequence matching. The warping path can end at any particular column of the last row, ranging from the index p to q., i.e. $\mathfrak{W}_k = (p,t); p \leq t \leq q$ and the warping path can start at any column, at the first row, i.e. $\mathfrak{W}_1 = (1,u); 1 \leq u \leq q$.

ii. **Continuity Condition** : The continuity condition of classical DTW is also not maintained by FSM: $\mathfrak{W}_k = (a,b)$ is followed by $\mathfrak{W}_{k+1} = (a',b')$; where $(a'-a) \in [0,1]$

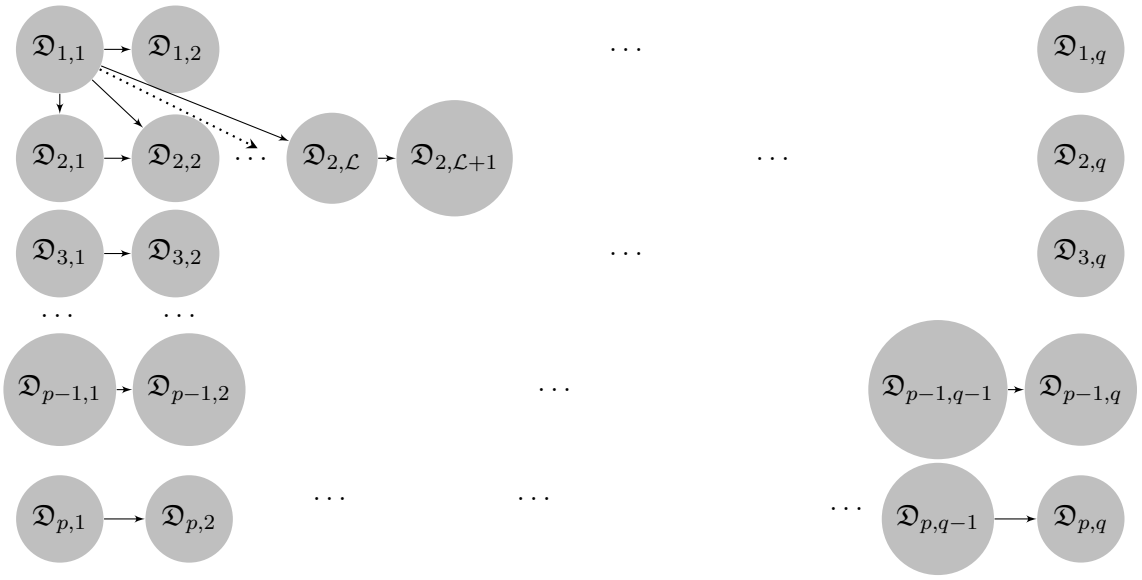


Figure 4.6: DAG constructed using a distance matrix obtained from two sequences.

and $(b' - b) \in [0, 1, ...|q - p| + 1]$. Due to this characteristic, FSM is able to jump outliers from the target signal.

iii. **Monotonicity Condition** : The monotonicity condition restricts the warping path from going back in time: $\mathfrak{W}_k = (a, b)$, $\mathfrak{W}_{k+1} = (a', b')$ is constrained by $(a' - a) \geq 0$ and $(b' - b) \geq 0$.

Depending on the requirement, the final distance can be normalized by two different ways.

i. The final distance stored in the cell $\mathfrak{P}_{p,\mathfrak{I}}$ can be normalized by the total number of correspondences. Let's say the normalized distance is denoted by $\bar{\mathfrak{d}}$, so $\bar{\mathfrak{d}} = \frac{\mathfrak{P}_{p,\mathfrak{I}}}{|\mathfrak{W}|}$; $|\mathfrak{W}| =$ The size of $\mathfrak{W}$ matrix.

ii. The other way of normalization is to divide by: total number of correspondence + total number of elements, skipped at the begging and end; i.e. $\bar{\mathfrak{d}} = \frac{\mathfrak{P}_{p,\mathfrak{I}}}{|\mathfrak{W}|+\{\{\mathfrak{W}(cnt-1,1)-1\}+\{q-\mathfrak{W}(1,1)\}\}}$ (please see line 30-31 of Algorithm 4)

4.2.2 Description of the Proposed Pseudo Code

As an output, the algorithm provides the path costs in $\mathfrak{P}$ matrix along with two other matrices ($\mathscr{R}$ and $\mathscr{C}$), which are used to backtrack the shortest path, giving the closest correspondence between query and target elements. The matrix $\mathscr{R}$ keeps track of rows and $\mathscr{C}$ keeps track of columns and the array $\mathfrak{W}$ merge these indexes to get the path. The overall process is given in Algorithm 4. The first row of the matrix $\mathfrak{P}$ is obtained by Eq^n 4.4a. All the other cells of $\mathfrak{P}$ are initialized with infinity. The cells of the path cost matrix ($\mathfrak{P}$) are calculated (refer to lines $5 - 26$) by iterating i over each row where k iterates over

Algorithm 4: FLEXIBLE SEQUENCE MATCHING

Input: $p(int), q(int), \mathfrak{D}_{pq}(double), \mathfrak{S}(float), \mathfrak{C}(float)$
Output: $\mathfrak{P}_{pq}(double), \mathfrak{W}(vector < double >)$

1 $\mathfrak{P} \leftarrow \infty$ ▷ Initialize all the cells of $\mathfrak{P}$ matrix by ∞
2 $elasticity \leftarrow |q - p|$ ▷ If the value of *elasticity* is not provided
3 **for** $j \leftarrow 1$ ***to*** q **do**
4 $\quad \mathfrak{P}(1, j) \leftarrow \mathfrak{D}(1, j)$ ▷ Fill the 1^{st} row of $\mathfrak{P}$ with the 1^{st} row values at $\mathfrak{D}$ matrix
5 **for** $i \leftarrow 2$ ***to*** p **do**
6 $\quad$ **if** $i = 2$ **then**
7 $\quad\quad R \leftarrow q$ ▷ Complete flexibility is given for choosing the 1^{st} node
8 $\quad\quad L \leftarrow 1$
9 $\quad$ **else**
10 $\quad\quad R \leftarrow min(q, (i-1) + elasticity)$
11 $\quad\quad L \leftarrow max(1, (i-1) - elasticity)$
12 $\quad$ **for** $k \leftarrow L$ ***to*** R **do**
13 $\quad\quad D \leftarrow ((k+1) + elasticity) - max(0, \{k - (i-1)\})$
14 $\quad\quad$ **for** $j \leftarrow k$ ***to*** D **do**
15 $\quad\quad\quad$ **if** $j = k$ **then**
16 $\quad\quad\quad\quad \mathscr{J} \leftarrow \mathfrak{C}$ ▷ Penalty for vertical links
17 $\quad\quad\quad$ **else if** $j = k + 1$ **then**
18 $\quad\quad\quad\quad \mathscr{J} \leftarrow 0$ ▷ No penalty for diagonal links
19 $\quad\quad\quad$ **else**
20 $\quad\quad\quad\quad \mathscr{J} \leftarrow \mathfrak{S} \times \{j - (k+1)\}$ ▷ Penalty proportional to number of jumps
21 $\quad\quad\quad$ **if** $\mathfrak{P}(i, j) > (\mathfrak{P}(i-1, k) + \mathfrak{D}(i, j) + \mathscr{J})$ **then**
22 $\quad\quad\quad\quad \mathfrak{P}(i, j) \leftarrow (\mathfrak{P}(i-1, k) + \mathfrak{D}(i, j) + \mathscr{J})$ ▷ Link between two rows
23 $\quad\quad\quad\quad \mathscr{R}(i, j) \leftarrow i - 1; \mathscr{C}(i, j) \leftarrow k$
24 $\quad\quad\quad$ **if** $\mathfrak{P}(i, j) > \mathfrak{P}(i, j-1) + \mathfrak{C} + \mathfrak{D}(i, j)$ **then**
25 $\quad\quad\quad\quad \mathfrak{P}(i, j) \leftarrow \mathfrak{P}(i, j-1) + \mathfrak{C} + \mathfrak{D}(i, j)$ ▷ Link from left node
26 $\quad\quad\quad\quad \mathscr{R}(i, j) \leftarrow i; \mathscr{C}(i, j) \leftarrow j - 1$
27 $\mathfrak{I} = \underset{p \le t \le q}{\text{argmin}}\ \mathfrak{P}(p, t)$ ▷ Column index of last row with the minimum value
28 $\mathfrak{r} = p; \mathfrak{c} = \mathfrak{I}; cnt = 1$
29 **while** $((\mathfrak{r} \ge 1) \& (\mathfrak{c} \ge 1))$ **do**
30 $\quad \mathfrak{W}(cnt, 1) = \mathfrak{r}; \mathfrak{W}(cnt, 2) = \mathfrak{c}$ ▷ Storing the cell indexes in the array for getting the warping path
31 $\quad \mathfrak{t} = \mathscr{C}(\mathfrak{r}, \mathfrak{c}); \mathfrak{r} = \mathscr{R}(\mathfrak{r}, \mathfrak{c}); \mathfrak{c} = \mathfrak{t}; cnt++$

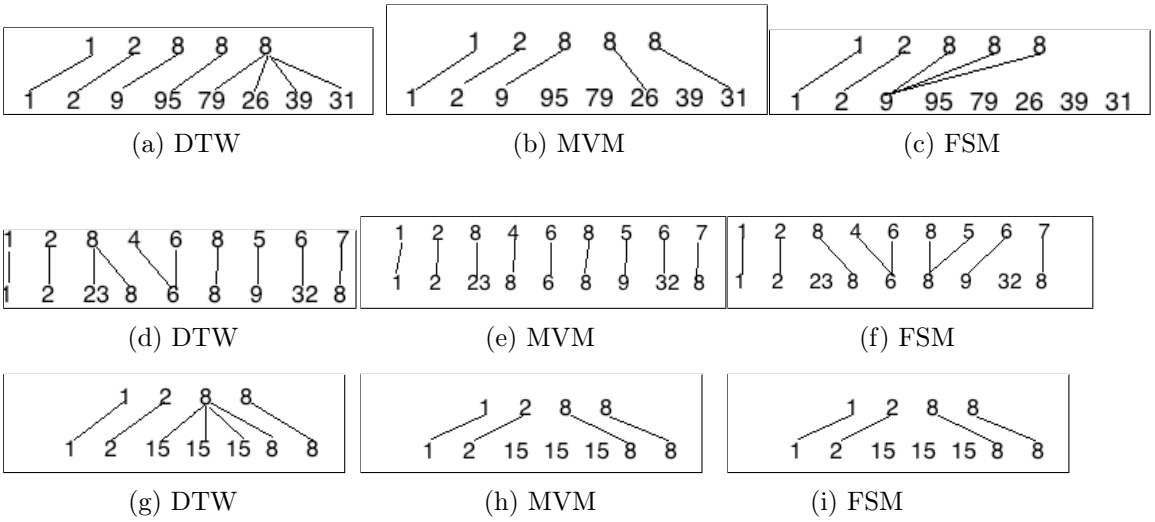
parent nodes at row (i-1) and j keeps track of child nodes. For calculating the cells of a row, firstly the range of parent nodes are calculated (line 5-11). When i = 2, parent nodes can be anywhere in first line to allow to skip irrelevant part (at no cost) at beginning of target. Next, for each parent node from L (left) to R (right), the possible connections with child nodes are calculated by j, which is iterated between k and D and takes into account,

many-to-one, one-to-many matching as well as jumps.

The multiple matching facility of FSM i.e. many consecutive elements of the query to one element of target (refer to Algorithm 4, line 14) and one to many (refers to Algorithm 4, line 24-27) is achieved along with the property of one to one matching. Note that, $jumpCost(\mathscr{J})$ is calculated based on number of skips already taken ($(j-(k+1))$ term in line 20 of Algorithm 4) and $skipCost(\mathfrak{S})$ (see line 20 of Algorithm 4). When the parent node at $(i-1,k)$ is connected with the node at $(i,k+1)$, no jumps are taken in this case, hence two consecutive elements of query are matched with two consecutive elements of target sequence. The process for calculating the warping path is shown in line 27-31. Initially the index of the column, ranging from p to q, containing minimum value at the last row of the path cost matrix is determined. Then a back tracked warping path ($\mathfrak{W}$) is obtained by iterating through the while loop.

4.2.3 Examples of matching with FSM

Using some toy examples, the behavior of FSM, compared to DTW and MVM is demonstrated in Fig. 4.7. The following Fig. 4.7a, 4.7b, 4.7b shows that, for the case of DTW, the 1^{st}, 2^{nd} and 3^{rd} elements of query are correctly matched with 1^{st}, 2^{nd} and 3^{rd} elements of targets, respectively. But the 4^{th} element (8) of query is forced to be matched with 4^{th} element (95) of target. The last element of query is also forced to be matched with other remaining elements of target (many-to-one matching), which significantly increases the final distance. MVM algorithm is able to improve the matching process by skipping outliers elements, ([95, 79]) from the sequence. But one can notice that the choice of outliers/inliers in questionable (elements 26 and 31 could also be considered as outliers considering values of query). This is mainly because of it's inability to have many-to-one matching and it's restriction that the numbers of matched elements from target sequence should be equal to number of elements in the query, compels the algorithm to match the 4^{th} and 5^{th} elements of Q with the 6^{th} and 8^{th} elements $(26, 31)$ of target. On the contrary, it can be easily visible that FSM can overcome these restrictions and can give right correspondence. In such case, small cost will minutely penalize the distance for one to many matching. In Fig.4.7d, we can see that DTW matching gives several wrong correspondences mainly due to it's inability to skip noisy elements. Due to the inability of having many to one correspondences and the hard constraint of finding same amount of elements from target sequence, with respect to the length of query sequence, MVM algorithm is also compelled to match with noisy elements. The FSM algorithm has overcome both of these bottlenecks and it is able to provide good matchings between the sequences, thanks to it's noise skipping ability. It is noteworthy to mention here that since the length of target and query signal are same, the *elasticity* for MVM and FSM algorithm is taken as 2, for having the benefits of outliers skipping from target signal. The clear interest of FSM and MVM over DTW can also be visible in Fig. 4.7g 4.7h 4.7i. While DTW matching drastically fails to find the right correspondences, MVM and FSM are able to give the right correspondences by skipping noisy elements.



(a) DTW (b) MVM (c) FSM

(d) DTW (e) MVM (f) FSM

(g) DTW (h) MVM (i) FSM

Figure 4.7: Study of partial matching, many-to-one and one-to-many matching and noise skipping abilities of (a)(d)(g) DTW, (b)(e)(h) MVM and (c)(f)(i) FSM.

4.2.4 Time complexity of FSM

The worst case complexity of FSM is calculated by computing total number of possible parent nodes at each row of $\mathfrak{P}$ matrix. In worst case, we would have a subset of at most $\{(2\times|q-p|)+1\}$ parent nodes (see line 10, 11 of Algorithm 4).) at each row of $\mathfrak{P}(i,j)(1 \leq i \leq p; 1 \leq j \leq q)$ matrix; which indeed can be represented as the vertices of DAG $\mathbb{G}$. Then for each of the parent nodes in row i is linked to at most $\mathscr{T}$ child nodes. Since there are total p rows, the algorithm complexity could be defined by: $[p \times \{(2 \times |q-p|)+1\} \times \mathscr{T}]$.

$$\mathscr{T} = [\{(j+1)+|q-p|\}-(j+1)]+1-(j-i)+1+1 = |q-p|+(j-i)+3 \approx |q-p|+3 \quad (4.5)$$

In Eq^n 4.5, the term $(j-i)$ can be ignored (it reduces the complexity) for performing worst case analysis. The term 3 comes from adding the diagonal link, link with the right child node on the same line as parent node and the child node just below parent node (see line 13 of Algorithm 4). The worst case complexity of FSM is $\Theta([|q-p|+3]\times[p\times\{(2\times|q-p|)+1\}]) = \Theta(2.p \times ((|q-p|+2)^2-1)) \approx \Theta(2.p \times (|q-p|+2)^2)$. This assumption is also true : $\Theta((2.|q-p|^2).p) < (\Theta(2.q^2.p)$. Furthermore the complexity can be reduced to linear $(\Theta((2.|q-p|^2).p) \approx \Theta(p)$ when $q \approx p$; i.e. matching whole target)). The time complexity of FSM can further be reduced by introducing an admissible lower bound. However, in this work we focus on demonstrating the utility of FSM; we will address speedup and index-ability of FSM in future work.

4.3 Generalization property of FSM

The architecture of FSM is general enough so that, with little modifications, it can be easily tuned to perform as other sequence matching algorithms.

4.3.1 Achieving DTW behavior from FSM (DTW-FSM)

The architecture of FSM can be easily tuned to perform as DTW, which can give completely identical results (see experiments in Section 4.4.1.1 and Fig.4.8a)[12]. The main idea here is to follow the same DP path as DTW and to restrain the skipping facility of FSM (see algorithm 5). The required changes are following :

i) Initialization portion (line 2 to 4) of Algorithm 4 is replaced by lines 2, 2.1, 3 and 4 in Algorithm 5. Here the *elasticity* and *skipCost* are taken as zero as in DTW, there are no possibility of skipping elements.

ii) The parent nodes are looped through all the columns of the array (line 6-11 of Algorithm 4 is replaced by line 4 in Algorithm 5). In DTW, the DP path is constructed through all the cells of every row of the dissimilarity matrix ($\mathfrak{D}$).

iii) Consider the sequence dissimilarity value from bottom right most cell of path cost matrix and backtrack the warping path from this cell and up to the top left most cell (replace the line 28 of Algorithm 4 by line 28 of Algorithm 5).

iv) The final dissimilarity value is normalized by dividing the path cost value at the bottom right cell of $\mathfrak{P}$ matrix by total number of correspondence between query and target sequence. The number of entries in warping path $\mathfrak{W}$, gives total number of correspondence.

To show that DTW-FSM performs identical as classical DTW, we performed experiments on GW dataset. As the GW dataset contains only segmented line information, so instead of classical DTW, we choose SSDTW for the experiments. Likewise, the P-R curve is compared against SSDTW-FSM[12] (refer to Section 4.3.4). Please note that, architecturally SSDTW and classical DTW are same and the only difference is in the process of calculating their respective warping paths. So, it is obvious that DTW vs. DTW-FSM would perform same as SSDTW vs. SSDTW-FSM and it is not necessary to show the separate plot of DTW vs. DTW-FSM.

[12] The proposed generalization property of FSM is demonstrated by plotting the curve only for GW dataset (not for other two datasets), by using first 2 queries (see Table 4.3) instead of total 15 queries. However, we maintained the same experimental protocol, mentioned in the last portion of section 4.4.1.1, i.e. all the occurrences are used for each of these 2 query images, for calculating mAP value.

Algorithm 5: DTW alike FSM

```
   Input: p, q, 𝔇, 𝔖, 𝔈
   Output: 𝔓, 𝔚
 1 𝔓 ← ∞
 2 elasticity ← 0; 𝔖 = 0; 𝔈 = 0 ▷ No skipping
2.1 𝔓(1,1) ← 𝔇(1,1)
 3 for j ← 2 to q do
 4     𝔓(1,j) ← 𝔇(1,j) + 𝔓(1,j−1)
 5 for i ← 2 to p do
10     R ← q
11     L ← 1
12     for k ← 1 to q do
13         D ← (k+1)
14         for j ← k to D do
               ....
20
23
26
▷ Warping path starts from bottom right most cell
28 𝔯 = q; 𝔠 = p; cnt = 1
   ....
   ....
```

Algorithm 6: MVM alike FSM

```
   Input: p, q, 𝔇
   Output: 𝔓, 𝔚
 1 ....
 2 ....
 5 for i ← 2 to p do
10     R ← max(1, {(i−1) + elasticity})
11     L ← min(q, (i−1))
12 for k ← L to R do
13     D ←
       (k+1+elasticity) − max(0, |k−(i−1)|)
14     for j ← (k+1) to D do
21         if 𝔓(i,j) > 𝔓(i−1,K) + 𝔇(i,j)
           then
22             𝔓(i,j) ← 𝔓(i−1,K) + 𝔇(i,j)
23             ℛ(i,j) ← i−1; 𝒞(i,j) ← k
27 𝔍 = argmin 𝔓(p,t)
       p≤t≤q
28 𝔯 = p; 𝔠 = 𝔍; cnt = 1
   .....
```

4.3.2 Achieving MVM behavior from FSM (MVM-FSM)

Attaining the MVM architecture from FSM, is quite easy since FSM is developed on the basis of MVM algorithm (see Fig.4.8a)[12]. Interested readers are requested to see [Latecki et al., 2007b] for detailed description on MVM. The required changes are as follows (see Algorithm 6): i) Restrict the limit of parent node (replace line 6-11 in Algorithm 4 by line 10-11 in Algorithm 6). ii) For each parent node, the child node always start from diagonal position (see the line 14 in Algorithm 6). iii) The portion, responsible for horizontal link of the dynamic programming (DP) path for FSM is removed here (the line 24-26 of Algorithm 4 are removed here). The final dissimilarity value is normalized here by dividing through the total number of query elements.

4.3.3 Achieving CDP alike behavior from FSM (M-CDP-FSM)

The continuous dynamic programming algorithm is described in Section 3.7.5 of Chapter 3. It can be visible from DP path of CDP that, more resistance is present for the diagonal link of the DP path compared to other two links. For our application, we investigated the effectiveness of this particular DP path. To do that, we changed the DP path in following manner and named this modified version as *Modified CDP* (M-CDP).

$$\mathfrak{P}(j,i) = \begin{cases} \infty & \text{if } 1 \le i \le p;\ j = 1,2 \\ \mathfrak{D}(j,i) & \text{if } i = 1;\ 3 \le j \le q \\ \min \begin{pmatrix} \mathfrak{P}(j-1,i-1) + \mathfrak{D}(j,i) \\ \mathfrak{P}(j-1,i) + \mathfrak{D}(j,i) \\ \mathfrak{P}(j,i-1) + \mathfrak{D}(j,i) \end{pmatrix} & \text{if } 1 < i \le p\ ;\ 3 \le j \le q \end{cases} \tag{4.6}$$

The output is obtained by $A(j) = \frac{1}{p}\mathfrak{P}(j,p)$. In such way, CDP uses same DP path as DTW and experiment (see experiments in Section 4.4.1.1 and Fig. 4.8b, Fig. 4.8c) shows minor declination of accuracy. So, it can be concluded that the original DP path of CDP along with it's respective weights has some positive impact.

CDP architecture [Oka, 1998] can also be achieved by the following modifications of FSM algorithm. Nevertheless, the original DP path of CDP can not be realized in this adaptive version of the algorithm. So, we used the modified version of CDP (M-CDP), which uses the simpler DP path of classical DTW. This allows us to compare modified CDP (M-CDP) with FSM turned into CDP alike behavior (see Section 4.4.1.1 and Fig.4.8c & 4.8b). The following modifications are performed in Algorithm 7) to achieve CDP alike behavior.

i) As for CDP, no skips are possible, so *elasticity* and *skipCost* are assigned to zero (see line 2).

ii) Contrary to other afore mentioned algorithms, for CDP architecture, we use target elements along rows and query elements along column wise (line 11 and 12).

iii) k is iterated over all parents (see line 12 in Algorithm 7 and notice that line 6-11 of Algorithm 4 are removed here).

iv) As, no skips are possible, so line 13 of Algorithm 4 is changed here.

v) Line 15-20 replaces the corresponding ones in Algorithm 4 to take into account the DP-Path of modified CDP.

vi) See the variable $\mathfrak{I}$, mentioned in line 30, stores the normalized distances, which are obtained by dividing the path cost values by the length of reference sequence (p). The minimum value of $\mathfrak{I}$ gives the indexes of sub-sequence (of target signal), which has optimal match with reference sequence and the final dissimilarity cost between the target and query sequences. So, there is no back track neither other distance normalization (lines 27-31 in Algorithm 4 are removed).

4.3.4 Attaining Subsequence DTW alike architecture from FSM (SSDTW-FSM)

FSM can also be easily modified to act like SSDTW algorithm. The detailed description on SSDTW is given Section 3.6.1 in Chapter 3). For generating *sub-sequence DTW* architecture from FSM, we simply follow the same technique mentioned in Section 4.3.1 by keeping into mind the aforementioned special constraint of *sub-sequence DTW*. The process of calculation of warping path is same as FSM and the rest of the part before the warping path calculation portion would remain same as in Algorithm 5. The final dissimilarity value ($\triangle(x,y)$ is normalized by dividing through the length of the warping path (same as in Algorithm 5).

Algorithm 7: CDP alike FSM

Input: $p, q, \mathfrak{D}, \mathfrak{S}, \mathfrak{C}$
Output: $\mathfrak{P}, \mathfrak{W}$

$\mathfrak{P} \leftarrow \infty$
$elasticity \leftarrow 0; \mathfrak{S} \leftarrow 0; \mathfrak{C} \leftarrow 0$
for $j \leftarrow 1$ ***to*** p **do**
  $\mathfrak{P}(1, j) \leftarrow \mathfrak{D}(1, j)$
for $i \leftarrow 2$ ***to*** q **do** // Target elements are considered row-wise in path cost matrix.
  for $k \leftarrow 1$ ***to*** p **do** // Query elements are considered column-wise[13] The loop for -
    $D \leftarrow min(k + 1, p)$ // - parent nodes are iterated for all query elements.
    for $j \leftarrow k$ ***to*** D **do** // Calculation of child nodes are same as DTW
      if $j = 1$ **then**
        $\mathfrak{P}(i, j) \leftarrow \mathfrak{D}(i, j)$ // 1^{st} column is copied from distance matrix.
      else
        if $\mathfrak{P}(i, j) > (\mathfrak{P}(i - 1, k) + \mathfrak{D}(i, j))$ **then**
          $\mathfrak{P}(i, j) \leftarrow (\mathfrak{P}(i - 1, k) + \mathfrak{D}(i, j))$
          $\mathscr{R}(i, j) \leftarrow i - 1; \mathscr{C}(i, j) \leftarrow k$
        if $\mathfrak{P}(i, j) > \mathfrak{P}(i, j - 1) + \mathfrak{C} + \mathfrak{D}(i, j)$ **then**
          $\mathfrak{P}(i, j) \leftarrow \mathfrak{P}(i, j - 1) + \mathfrak{C} + \mathfrak{D}(i, j)$
          $\mathscr{R}(i, j) \leftarrow i \; \mathscr{C}(i, j) \leftarrow j - 1$
  $\mathfrak{I}(i, 1) = \mathfrak{P}(i, k)/k$

4.4 Experimental Evaluation

The experimental section is divided into two categories: *i) Line segmentation based word spotting*, *ii) Pseudo-word segmentation based word spotting*. Depending on the characteristics of the documents, it can be comparatively easier to perform word segmentation or line segmentation. For example, if inter word gap is not significant enough (e.g. old historical documents), word segmentation can be very difficult. This segmentation difficulty is highly script, writer and language dependent. In many cases, results of line segmentation can be better than the results of word segmentation especially for old manuscripts. The advantage of line segmentation based approach is that it can be possible to spot hyphenated words spanned into two lines. In most of the languages, an acceptable line segmentation result can be obtained by simple *"pixel projection along the text line direction"*. But, line segmentation is also difficult, when there is a high amount of warping, slant or high degradations. In such cases, line segmentation could be more difficult than word segmentation. Moreover, thanks to the properties of proposed FSM techniques, it does not

need perfect word segmentation. So, FSM is able to spot words on pieces of lines/words (pseudo words). Thus due to these pros and cons of both segmentation techniques, we have performed the experimental evaluations on both segmentation approaches, to evaluate our proposed technique *i)* on segmented lines *ii)* on pieces of lines/pseudo words.

4.4.1 Results on Segmented Lines

The experiments in this category are performed on two historical handwritten, datasets; *George Washington(GW)* and *Japanese* dataset. GW dataset has 675 text lines. These well segmented text lines are obtained from [Terasawa and Tanaka, 2009]. The *Japanese* dataset comes from the manuscripts of *"Akoku Raishiki (The diary of Matsumae Kageyu)"*.

4.4.1.1 Results on GW dataset

To perform the experimentation with *George Washington* dataset, 15 query images are used (see Fig. 4.9a). These are the same query images, as the one already used in [Leydier et al., 2007], [Terasawa and Tanaka, 2009] (see Table.4.3). We used the same segmented lines, ground truth and slit style based HOG feature values for these two datasets as the one used in [Terasawa and Tanaka, 2009]. These segmented lines are pre-processed for noise removal, skew and slant correction. All of these data are availed by the authors of [Terasawa and Tanaka, 2009]. The protocol for accuracy calculation, is minutely different for our case in comparison with the one mentioned in [Terasawa and Tanaka, 2009]. Consequently, the only difference between results from [Terasawa and Tanaka, 2009] and ours comes from the matching algorithm (CDP[14] [15] vs. FSM) and from the difference in accuracy calculation protocol. Most probably these following mentioned reasons are responsible for this difference in results of their implementation of CDP based word spotting system and our implementation of CDP based system: *i)* No detailed information is provided in [Terasawa and Tanaka, 2009], about the used experimental protocol for calculating mAP values. We think, it is different than the one, we used. In our case, we used all the occurrences of each query words as queries (explained below) for calculating mAP values (for the Japanese dataset, we used 10 occurrences of every query words). But in [Terasawa and Tanaka, 2009], we do not have any information about their experimental setup. *ii)* We used different strategy for handling multiple occurrences of a query word in line (explained below), but we don't have any information regarding this, for the one mentioned in [Terasawa and Tanaka, 2009].

The ground truth data contains image name and location of each query word. A single query word can appear several ways in the image: *i)* particular query image can occur in a line of the document; *ii)* a query image can occur multiple times in a single line; *iii)* the query can occur as a hyphenated word, where the two parts of the query will occur in two consecutive text lines; *iv)* there can be a line, which can contain complete query word and a part of hyphenated query word (refer to Table 4.3). The issue of having multiple query

[14]The implementation of CDP, used here is taken from: `http://www.diva-portal.org/smash/get/diva2:347724/FULLTEXT01.pdf`. Page no. 86

[15]For our version of the implementation of CDP, please visit: `http://continuousdynamicprog.blogspot.in/`

Table 4.3: Statistics of the query words in GW dataset

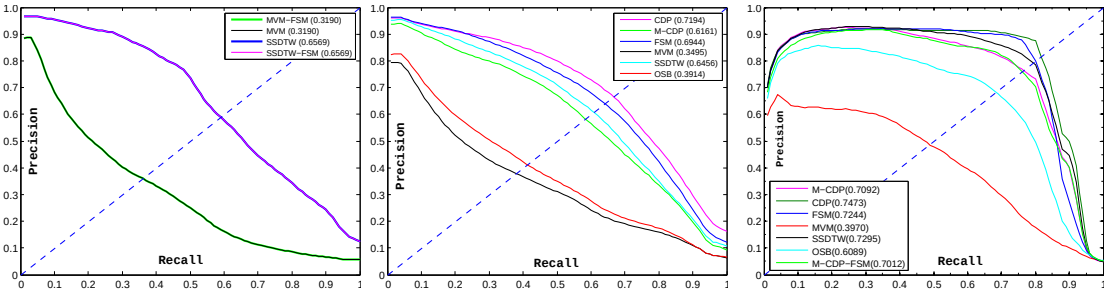
	$\breve{Q}^{\circ}$	$\widetilde{N}^{\circ}(\hat{N}^{\circ})$	$\bar{C}$	$\bar{F}$
GW Dataset	1755	37 (0)	0.701	0.592
	Company	20 (1)	0.737	0.662
	Cumberland	13 (1)	0.818	0.845
	December	26 (0)	0.755	0.761
	Fort	22 (0)	0.689	0.660
	Instructions	21 (1)	0.955	0.958
	Letters	22 (0)	0.841	0.884
	October	15 (1)	0.705	0.713
	Orders	33 (0)	0.601	0.568
	Recruits	12 (0)	0.837	0.741
	Sergeant	12 (0)	0.731	0.799
	Virginia	14 (0)	0.515	0.495
	Winchester	15 (4)	0.693	0.674
	Captain	27 (1)	0.660	0.646
	Regiment	17 (0)	0.546	0.411

words (hyphenated or complete) in one line can only be handled by CDP. The architecture of CDP is capable to spot multiple query words in a single line, if a good distance threshold (difficult to estimate and highly data dependent) can be estimated, which can distinguish between right and wrong matches.

In the case of hyphenated words, it is obvious that the word will occur in two consecutive lines. So, in that case, two consecutive lines are merged and their corresponding feature values are also merged. When one particular line is having more than one query word, e.g. a line has n occurrences of query word, the line image is considered n times for matching with the particular query image. For each consideration, the pixels of only one occurrence is kept unchanged in the line image and the pixel intensities of other occurrences exists in the same line is changed into zeros to avoid ambiguity in matching process. This process is repeated for each occurrences in that particular line. For the fair comparison with all other algorithms (including FSM) against CDP, we need to do this restructuring of the lines, having multiple occurrences of query words. When a line has one or more than one complete word and a hyphenated word, the consecutive two lines are merged for matching the hyphenated word with the query word image. Then, the particular line having multiple occurrences of complete query image among these two consecutive lines is identified. Similar technique as previous one is followed for this particular line by keeping the pixels of one occurrence unchanged and changing the intensities of the pixels of other occurrences, including the portion of hyphenated word. For obtaining the accuracy, we calculate the mean average precision (mAP)[17] for all of the selected 15 query words (see Fig.4.8). Let's denote the query word by $\mathfrak{q}_s^t$ where s represents all the considered query words for the experiment $1 \leq s \leq 15$ and t represents the number of occurrences of each of the query words in complete GW dataset. For example, the particular query word *"Winchester"* has 15 occurrences in the form of full word (not hyphenated) and has 4 occurrences in the form of hyphenated words, so $1 \leq t \leq 19$ (see Table 4.3).

[16] Please see Section 4.3 for details.

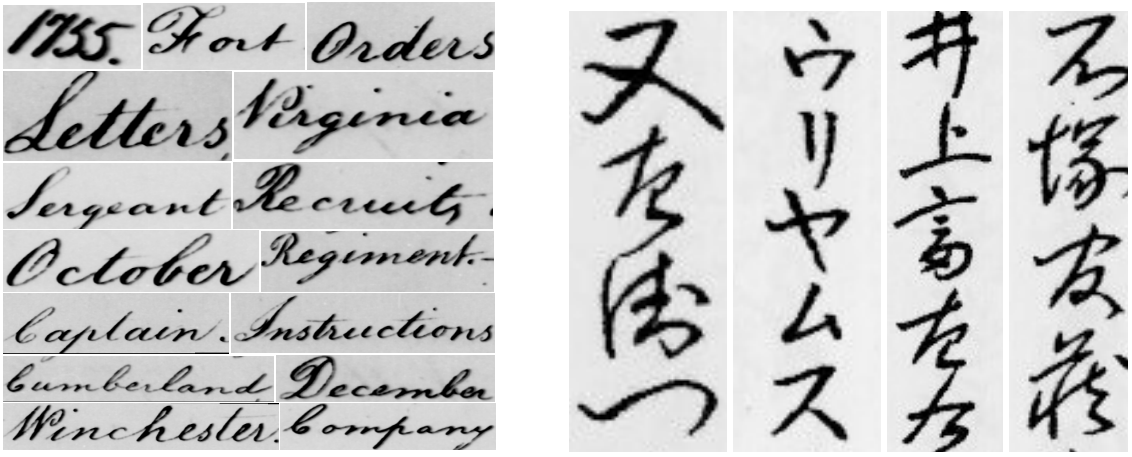
[17] https://en.wikipedia.org/wiki/Information_retrieval#Mean_average_precision



(a) (b) (c)

Figure 4.8: a) Performance comparison of MVM with MVM-FSM[16]and of SSDTW with SSDTW-FSM[16] (see digitized version of the image, as the curves are completely overlapped). b) Performance comparison of CDP, M-CDP[16], FSM, MVM, SSDTW on GW dataset. c) Performance comparison of CDP, M-CDP, M-CDP-FSM[16], FSM, MVM, SSDTW, OSB on Japanese dataset

We consider all the occurrences of each of the 15 query words for the experiment, e.g the accuracy is computed on all the 19 (complete-15, hyphenated-4) occurrences of the query word (*"Winchester"*). In simple words, we calculated the average precision over $(37 + 21 + 14 + 26 + 22 + 22 + 16 + 33 + 12 + 12 + 14 + 19 + 28 + 17) = 293$ query images. The accuracy of each query word is given in Table 4.3. The mean precision and recall (P-R) are also calculated and plotted over all of the 15 query words (see Fig.4.8). The average mAP of each algorithm over 15 query words are mentioned inside the legend of the graph. It can be seen from the graphs and from mAP values that FSM has outperformed all other classical sequence matching technique except CDP. There are slight difference in mAP $(0.7194 - 0.6944 = 0.025)$ between CDP and FSM.



(a) Query images used for GW dataset (b) Query images used for Japanese dataset

Figure 4.9: Query images of (a) GW dataset (b) Japanese dataset.

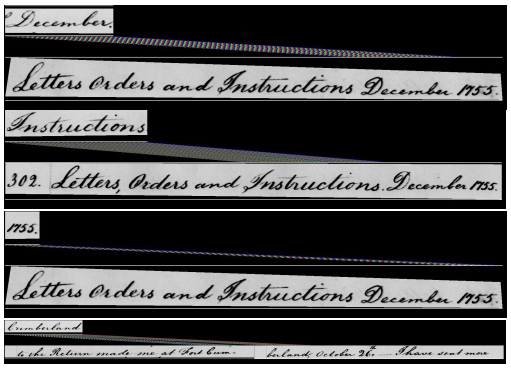
Table 4.4: Statistics of the query words in Japanese dataset

	$\breve{Q}^{\circ}$	$\widetilde{N}^{\circ}(\hat{N}^{\circ})$	$\bar{C}$	$\bar{F}$
Japanese	A.Matazaemon	154 (10)	0.841	0.820
	B. Uriyamusu	68 (5)	0.821	0.763
	C.InoueTomizou	22 (3)	0.722	0.717
	D.IshizukaKanzou	21 (4)	0.603	0.596

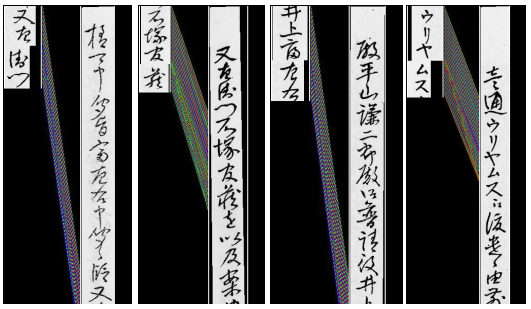
4.4.1.2 Results on Japanese dataset

For the case of Japanese dataset, same features and aforementioned experimental protocol is used. The experimented query images of this dataset are shown in Fig. 4.9b. We also perform the same aforementioned restructuring of lines for handing different types of occurrences of query words in a line. The mAP of this dataset is calculated over 4 query images. Compared to GW dataset, there are many occurrences of each query words in the Japanese dataset. So, to reduce the computation time to calculate mAP, we randomly choose 10 occurrences of each query word. Among these 10 occurrences, we took 5 occurrences of complete words and other 5 occurrences of hyphenated words. If any query word has less than 5 occurrences of hyphenated words (e.g. C.InoueTomizou; see Table 4.4), then we choose more complete words instead of hyphenated words. Same as GW dataset, the computed P-R values for Japanese dataset can be visible in Fig. 4.8c. Also for this dataset, FSM has outperformed all other classical sequence matching technique except CDP and there exist minor difference in accuracy ($0.7473 - 0.7244 = 0.022$).

By analyzing the results of both the dataset, we can say that, most probably, the special DP path and associated weights with the DP paths of CDP are responsible for this slightly better result (see the result of M-CDP, (where instead of the complex DP path, used in classical CDP, we used a simple DP path, as the one used in classical DTW.) is lower than original CDP). This rationale can be cross checked by seeing the performance of M-CDP-FSM on GW dataset, where simple DP path (with no weight, e.g. classical DTW) is used, by keeping the same CDP's architecture. This modification helps us to understand the effect of the special DP path of CDP. It can be seen from Fig.4.8b, that the performance of M-CDP-FSM is lower than CDP. Whereas due to the noise or outliers (derivatives also) skipping capability, FSM can perform well (see some visual examples, shown in Fig 4.11d.) in the presence of noise and derivatives (with comparative high variations). During the experiments, we have observed that the noise skipping ability of FSM is an advantageous characteristic over CDP. But when there are no or little noise present in the signal, FSM encounters some problem to distinguish between inlier and outliers and in these cases CDP performs comparatively better than FSM. As GW and Japanese datasets does not contains much noise, so the overall statistical accuracy of FSM is slightly lower than CDP. Nevertheless, for some query words in GW dataset, FSM has outperformed CDP (the highlighted words in Table. 4.3). This is mainly due to the outliers skipping ability and multiple matching (many-to-one and one-to-many) capability of FSM over CDP, and GW dataset is comparatively more noisy than Japanese dataset, which is very less noisy. In the following Fig. 4.10, we have give some visual examples of matching query words in segmented lines with FSM.



(a) Four separate query words are matched with corresponding segmented lines in GW dataset



(b) Four separate query words are matched with corresponding segmented lines in Japanese dataset

Figure 4.10: Visual examples of matching in (a) GW dataset (b) Japanese dataset.

4.4.2 Results on Segmented Pseudo Words

The application of FSM on segmented pseudo words is performed on CESR dataset[18]. The details of this dataset[19] is described in Section 2.5.4 of Chapter 2. For performing the experiments, we manually selected 123 queries, based on number of occurrences, their possible relevant derivatives and their meaningfulness as queries. To perform the experiment with these queries, we followed the architecture and framework, described in Fig. 2.2 of Chapter 2. For this dataset, the column based features (see Table 2.2 in Section 2.3.2.1 of Chapter 2) are extracted and used for matching. The queries and their corresponding considered derivatives, are categorized into 10 groups (see Table 4.5). The target set for each query is composed of the pages from the book, where the query has appeared. The experiments are performed in group-wise manner. The words, enclosed by rectangle box, represents each group. Each and every query word of a group are considered as query. When one word from a group is considered as query the other words of that particular group are considered as its derivatives.

The accuracy for this dataset is calculated by averaging the accuracies over all of the images (all images are used as query) in each group. One should notice that, due to improper segmentation, some of the target words can be smaller than considered query word. In such cases, that particular small target words are considered as queries and the original query word is considered as target word and the 2^{nd} way of normalization (see bottom portion of Section 4.2.1) is used for FSM, to avoid wrongly segmented small words from appearing at top ranked positions in overall nearest neighbor ranking process. Moreover, very small segmented words (e.g. one or two characters) are pruned by considering their width with respect to the width of original query word. We heuristically decided, if the width of any target word is lesser than 35% of the width of query word, we pruned the target word.

It is evident from the result, mentioned in Table 4.5 that in many cases FSM has

[18]http://cesr.univ-tours.fr/

[19]The complete dataset is publicly available at `https://github.com/tanmayGIT/CESR_DataSet`

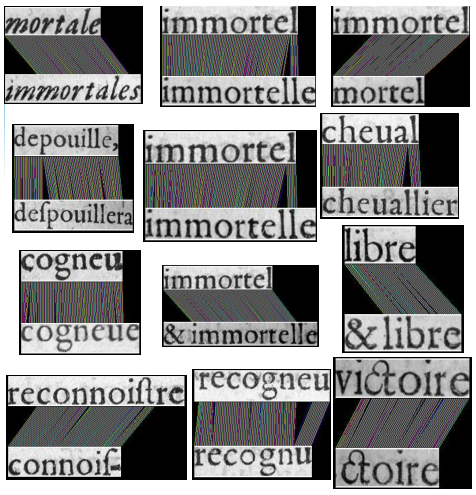
$\breve{Q}^\circ$	$\widetilde{N}^\circ$	$\widetilde{O}^\circ$	$\bar{C}$	$\bar{F}$
liberte	2	219	0.308	0.230
libre	19	2511	0.712	0.679
librement	3	462	0.363	0.374
TW = 3192; $\bar{AC}$ = 0.461 $\bar{AF}$ = 0.428				
cheualier	4	522	0.501	0.470
cheual	38	5099	0.509	0.443
cheualerie	1	171	0.451	0.464
cheuallier	1	168	0.421	0.426
cheuaus	4	546	0.395	0.438
cheuaux	8	1064	0.400	0.334
TW = 7570; $\bar{AC}$ = 0.446 $\bar{AF}$ = 0.429				
victoire	23	3173	0.646	0.577
victo	1	125	0.593	0.511
victoi	2	325	0.542	0.368
victoire	23	3173	0.646	0.578
victorieus	5	523	0.375	0.337
victorieux	6	901	0.518	0.486
TW = 8220; $\bar{AC}$ = 0.553 $\bar{AF}$ = 0.476				
mortel	11	1032	0.398	0.403
immortales	1	166	0.132	0.112
immortalit	8	968	0.332	0.290
immortel	14	930	0.200	0.181
immortelle	10	609	0.221	0.185
immortels	2	168	0.227	0.196
mort	28	2934	0.087	0.106
mortale	4	152	0.155	0.141
mortales	1	29	0.131	0.136
mortalia	5	275	0.107	0.074
mortalibus	2	285	0.078	0.044
mortalite	1	176	0.225	0.216
morte	10	1033	0.078	0.071
mortelle	9	822	0.285	0.290
mortelles	8	1172	0.264	0.228
mortels	4	324	0.227	0.225
mortem	5	243	0.036	0.026
mortes	1	169	0.054	0.101
morgue	2	217	0.026	0.024
morti	1	156	0.200	0.156
mortis	3	357	0.044	0.039
morts	3	221	0.082	0.099
mortua	1	184	0.031	0.025
mortuum	1	173	0.095	0.061
mortuus	1	184	0.030	0.024
mortz	3	539	0.076	0.092
TW = 13518; $\bar{AC}$ = 0.147 $\bar{AF}$ = 0.136				

$\breve{Q}^\circ$	$\widetilde{N}^\circ$	$\widetilde{O}^\circ$	$\bar{C}$	$\bar{F}$
roy	119	12063	0.272	0.293
royales	1	154	0.066	0.059
royaume	2	263	0.646	0.765
royaumes	4	485	0.747	0.763
royaus	1	161	0.116	0.114
royaute	2	274	0.188	0.168
royaux	1	191	0.429	0.357
royne	8	915	0.252	0.193
roys	17	1442	0.173	0.158
rois	16	1894	0.021	0.040
roistre	1	110	0.012	0.013
roit	20	2744	0.017	0.038
TW = 20696; $\bar{AC}$ = 0.245 $\bar{AF}$ = 0.247				
reconneut	1	180	0.045	0.045
reconnoi	1	173	0.047	0.086
reconnoissable	2	331	0.343	0.293
reconnoissanc	6	937	0.318	0.318
reconnoissant	1	171	0.334	0.294
reconnoissent	1	174	0.307	0.103
reconnoissoier	1	175	0.219	0.175
reconnoissoit	2	338	0.291	0.281
reconnoissons	3	486	0.344	0.277
reconnoistre	7	975	0.140	0.132
reconnoit	5	723	0.052	0.092
reconnoitre	4	634	0.162	0.160
reconnoy	3	360	0.088	0.083
reconnu	2	286	0.044	0.097
reconnue	1	174	0.037	0.045
TW = 6117; $\bar{AC}$ = 0.185 $\bar{AF}$ = 0.165				
reconoissent	1	178	0.092	0.083
reconoissoit	1	167	0.104	0.095
reconoitre	1	175	0.055	0.049
recogneu	1	146	0.143	0.134
recognois	1	182	0.328	0.335
recognoissan	1	180	0.312	0.255
recognoissanc	3	506	0.330	0.330
recognoissant	1	166	0.297	0.249
recognoisse	1	29	0.439	0.380
recognoissons	1	163	0.426	0.383
recognoistre	8	972	0.118	0.111
recognoit	1	170	0.343	0.309
recognoitre	2	318	0.150	0.139
recognu	1	172	0.148	0.135
TW = 3524; $\bar{AC}$ = 0.235 $\bar{AF}$ = 0.213				

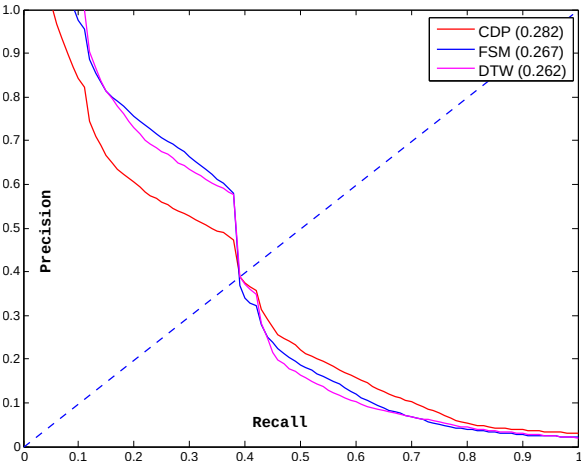
$\breve{Q}^\circ$	$\widetilde{N}^\circ$	$\widetilde{O}^\circ$	$\bar{C}$	$\bar{F}$
connoi	4	510	0.019	0.024
connois	5	821	0.312	0.321
connoissan	11	1053	0.475	0.395
connoissance	34	4665	0.456	0.456
connoissans	2	290	0.224	0.167
connoissant	4	536	0.482	0.390
connoisse	1	166	0.505	0.489
connoissent	4	589	0.431	0.397
connoissoit	3	378	0.459	0.475
connoist	1	116	0.276	0.287
connoistre	12	1531	0.223	0.186
connoit	5	688	0.035	0.047
connoitre	3	381	0.080	0.086
connoy	5	608	0.021	0.035
connu	3	404	0.018	0.023
connue	1	166	0.031	0.025
connues	4	350	0.029	0.025
connus	1	169	0.018	0.024
TW = 13431; $\bar{AC}$ = 0.228 $\bar{AF}$ = 0.214				
cognoissons	1	181	0.505	0.482
cognoistre	12	1705	0.583	0.557
cognoit	6	915	0.147	0.113
cognoitre	2	266	0.371	0.407
cognosci	1	159	0.142	0.205
cogneu	3	518	0.098	0.100
cogneue	2	298	0.065	0.077
cogneut	1	162	0.121	0.088
cognoi	1	84	0.094	0.074
cognois	5	753	0.519	0.537
cognoisoit	1	121	0.569	0.555
cognoissan	2	320	0.516	0.539
cognoissance	29	4187	0.578	0.552
cognoissances	1	168	0.716	0.625
cognoissant	1	189	0.486	0.480
cognoissent	6	657	0.492	0.459
cognoissions	1	174	0.519	0.555
cognoissoint	1	167	0.499	0.502
TW = 11024; $\bar{AC}$ = 0.390 $\bar{AF}$ = 0.384				
despouille	4	355	0.721	0.701
despouiller	5	771	0.785	0.757
despouillera	1	152	0.763	0.768
depouille	1	181	0.754	0.719
depouiller	1	188	0.392	0.403
TW = 1647; $\bar{AC}$ = 0.683 $\bar{AF}$ = 0.670				

Table 4.5: Statistics and results of CESR dataset. The words, enclosed by rectangular box, represents each group.

The target set of words ($\widetilde{O}^\circ$) against each query is generated only from the pages, where the query word has appeared. $\breve{Q}^\circ$ = Queries. $\widetilde{N}^\circ$ = No. of occurrences of the query word. $\bar{C}$ = mAP of CDP; $\bar{F}$ = mAP of FSM; TW = Total words in target set; $\bar{AC}$ = average mAP of CDP; $\bar{AF}$ = average mAP of FSM.



(a) Example of matching by FSM



(b) Comparative performance of CDP, FSM and DTW

Figure 4.11: Comparative results of DTW, CDP and FSM. Some visual examples of CESR dataset along with the matching provided by FSM (the top image is the query and bottom one is target).

outperformed CDP, whereas in several other cases the accuracy by FSM and CDP are quite similar (see the highlighted rows in Table 4.5). There are some queries, on which CDP has outperformed FSM. Most probably, thanks to the special DP path and it's associated weights, which has helped CDP to perform better than FSM. CDP is more effective than FSM, when there are less presence of noise inside the word images and there are less variations between query and it's considered derivatives. Although the average performance (on 123 query words) of CDP is better than FSM but the special characteristics of FSM can be highly helpful in various domains of time series sequence matching e.g. finance, video retrieval, shape matching [Latecki et al., 2007b, Jeong et al., 2011, Albrecht, 2009, Latecki et al., 2007c] etc.

4.4.2.1 Results on UCR dataset

Thanks to the generalized properties of FSM, it can be applied in typical time series matching problems. To demonstrates the applicability of FSM on typical time series domain, we applied it on various time series data, obtained from UCR archive. For the details about this dataset, please see Section 1.4 of Chapter 1. The results are given in terms of error rates, which is defined as :

$$\text{Error rate} = \frac{(\text{total number of testing data}) - (\text{total number of correctly classified data})}{\text{total number of testing data}} \tag{4.7}$$

The results of DTW, mentioned in the website of UCR archive (DTW-UCR)[20] are consider for comparison. Along with their version of DTW (DTW-UCR), we also tested our implementation of DTW. Although in most of the cases these two results are identical (DTW-UCR v/s DTW) but in some cases, there exists minor difference between both of these values. CDP [Oka, 1998] was originally proposed for the domain of speech recognition, but it had never been applied for time series classification, more precisely on UCR dataset. As far our knowledge is concerned, we are the first ones to investigate it on UCR dataset. It can be visible from the Table. 4.6 that in several cases CDP has outperformed DTW and FSM (highlighted ones). Moreover, we also applied FSM on some of these datasets for analyzing it's performance. It can be visible that FSM has not performed well but please note that this is a preliminary stage of investigation of FSM in the domain of time series classification. Calculation of jump costs, plays a vital role in the performance of FSM. Currently jump cost is calculated each time while two sequences are compared. For classification task, this is not the right way of calculation of jump costs. Jump cost should be calculated from the training set of each data (available in UCR archive), then this single value of jump cost should be applied on entire test set. But, more investigations and future work are required in this direction.

4.5 Conclusion

In this chapter, we presented a new robust sequence-matching algorithm called as FSM algorithm, which can be easily modified into the architecture of other sequence matching techniques e.g. DTW, SSDTW, MVM, and CDP etc. The ability to skip outlier elements present at any position of the target sequence, and the facility for many-to-one and one-to-many matching, makes the proposed FSM algorithm robust, generalized and applicable for various domains of time series sequence matching e.g. finance, video retrieval, shape matching [Latecki et al., 2007b], [Jeong et al., 2011], [Albrecht, 2009], [Latecki et al., 2007c] etc. In this research work, we have successfully demonstrated the usefulness of the proposed FSM algorithm in the domain of word spotting at line level as well as incorrectly segmented word level. We plan to perform more experiments to evaluate the robustness of FSM algorithm in comparison with other approaches on more bigger and multilingual datasets also in general time series database. An extension of FSM, which is able to skip noisy elements in query is presented in the next chapter. In future, we would also like to work on the reduction of time complexity of FSM algorithm.

[20] `http://www.cs.ucr.edu/~eamonn/time_series_data/`

Table 4.6: Summary of classification performance (error rate) on the datasets, obtained from UCR archive

Dataset	DTW-UCR	DTW	CDP	FSM
50words	0.31	0.30989	0.197802	
Adiac	0.396	0.396419	0.475703	
Beef	0.5	0.5	0.466667	0.5
CBF	0.003	0.00333333	0	0.10778
Coffee	0.179	0.178571	0.214286	0.25
ECG200	0.23	0.23	0.16	0.11
FaceAll	0.192	0.24497	0.202367	
FaceFour	0.17	0.170455	0.113636	0.159091
fish	0.0167	0.165714	0.102857	
Gun_Point	0.093	0.0933333	0.0333333	0.046667
Linghting2	0.131	0.131148	0.131148	0.180328
Linghting7	0.274	0.287671	0.260274	0.287671
OliveOil	0.133	0.133333	0.166667	0.166667
OSULeaf	0.409	0.409091	0.305785	
SwedishLeaf	0.21	0.208	0.176	
synthetic_control	0.007	0.0333333	0.013333	0.12
Trace	0	0.01	0.01	0.24
Two_Patterns	0	0.00175	0.00075	
wafer	0.02	0.0201168	0.0162232	
yoga	0.164	0.164333	0.147667	
ChlorineConcentration	0.352	0.351562	0.377604	
CinC_ECG_torso	0.349	0.349275	0.254348	
Cricket_X	0.223	0.223077	0.2	
Cricket_Y	0.208	0.207692	0.146154	
Cricket_Z	0.208	0.207692	0.182051	
DiatomSizeReduction	0.033	0.0326797	0.0457516	
ECGFiveDay	0.232	0.232288	0.198606	0.215777
FacesUCR	0.0951	0.0951219	0.0453659	
Haptics	0.623	0.623377	0.587662	
InlineSkate	0.616	0.616364	0.598182	
ItalyPowerDemand	0.05	0.0495627	0.0660836	0.0379001
MALLAT	0.066	0.0660981	0.0511727	
MedicalImages	0.263	0.263158	0.234211	
MoteStrain	0.165	0.165335	0.113419	
SonyAIBORobotSurface	0.275	0.274542	0.222962	
SonyAIBORobotSurfaceII	0.169	0.16894	0.147954	
StarLightCurves	0.093			
Symbols	0.05	0.0502513	0.0251256	
TwoLeadECG	0.096	0.095698	0.0834065	

Chapter 5

Exemplary Sequence Cardinality

Contents

Abstract

In this chapter, a new sequence matching algorithm called as Exemplary Sequence Cardinality (ESC) is proposed. ESC is an extension of FSM. It has all the qualities as FSM, in addition, ESC has the ability to skip the elements from query e.g. LCSS, OSB. In case of word spotting application, the outliers skipping capability of ESC makes it less sensible to local variations in the spelling of words, and also to noise present in the query and/or in the target word images. By experimenting on printed historical document images, we have demonstrated the interest of proposed ESC algorithm in specific cases when incorrect word segmentation and word level local variations occur regularly. Thanks to it's outliers skipping facility from query sequence, this technique gives more flexibility to user for choosing query images. This facility helps users to evade from rigorous searching for perfect query.

5.1 Introduction

The general introduction and literature review on sequence matching techniques are mentioned in previous chapter on *Flexible Sequence Matching*. Continuing from this introduction, in this section, we would like to directly introduce the proposed algorithm; *Exemplary Sequence Cardinality* [Mondal et al., 2015a]. ESC is the extended version of our previ-

Table 5.1: Comparison of characteristics of different sequence matching techniques

Method Name	**Continuity (Skipping)**	**Boundary Condition**	**Nature of Correspondence**
FSM	Each query elements has to be matched with some target elements but not vice versa. Skipping is possible at any position of **target signal** (skipping penalty is added).	Same as MVM.	**One-One, Many-One, One-Many**
ESC [Proposed]	Best optimal correspondences are obtained between query and target sequences. Skipping is possible at any position of **query and target signal** (skipping penalty is added).	Same as OSB and LCSS.	**One-One, Many-One, One-Many**

ously proposed sequence matching algorithm; Flexible Sequence Matching (FSM) [Mondal et al.,]. We would like to remind you that FSM has the following qualities : *i)* can perform one-to-many, many-to-one, one-to-one matching. *ii)* can skip outliers, belonging at any position in the target i.e. at the beginning, at the end or in between the inlier. Hence, FSM is able to find a partial sequence corresponding to a query inside a longer target sequence. But notably, FSM is not able to skip outliers from the query along with all the properties mentioned above. For some application, it is evident that there can be high possibility or could be an easier option to use an inexact query image. This is essentially true in the query-by-example category of word spotting system, for which rigorously searching for a perfect query image could be difficult. Hence in ESC, we tried to overcome this particular drawback of FSM and proposed an improved version of this algorithm. In ESC, to have the property of skipping noise from query along with all other properties of FSM, the algorithmic architecture of FSM is highly modified (details are given in section 5.2). The property wise difference between ESC & FSM is similar to the difference between OSB & MVM.

The proposed system, is more robust to *i)* degradation noise *ii)* word derivatives *iii)* improper segmentation issues. Regarding derivatives, there can exists several variants or variations of some words. For example, in French, the word *cheval (horse)* can have derivatives like *"chevaux"*,*"chevalerie"*, *"chevalier"*. In old French, other derivates also exist due to lexical variations: *"chevallier"*, *"chevaus"*; and also the "*v*" is often printed as a "*u*". There exist some approaches for word spotting, which can spot words by skipping

the prefix and suffix of segmented words [Terasawa and Tanaka, 2009, Mondal et al.,]. But the proposed sequence matching based word spotting approach is also more robust to spelling variations.

5.2 Exemplary Sequence Cardinality

We describe here the complete mathematical architecture of ESC algorithm, which has an initial resemblance with the FSM architecture. ESC creates a relation $\mathbb{R}$ from two finite sequences x (query) and y (target) of different lengths p and q: $x = (x_1, x_2, ..., x_p)$; $y = (y_1, y_2, ..., y_q)$. The goal of the algorithm is to find $x^{'}(x^{'} \subset x)$ such that $y^{'}(y^{'} \subset y)$ best matches with each others. The correspondence can be thought as a relation on the set of indices $1, ...p \times 1, ...q$, where, one-to-one, one-to-many and many-to-one mapping are also included. After knowing the correspondence, it is easy to compute the distance between the two sequences by maintaining order preserving relation $\mathbb{R}$ between x and y.

5.2.1 Theoretical Description

First the difference matrix[1] $\mathfrak{D}$ between elements of the two sequences is calculated by using Euclidean distance: $\mathfrak{D}_{i,j}=\sqrt{(y_j - x_i)^2}$; $1 \leq i \leq p; 1 \leq j \leq q;$. By following the same process as FSM, the difference matrix $\mathfrak{D}$ between elements of the two sequences is calculated and also the cost function $\mathcal{H}$ is defined in the same manner as FSM for constructing the DAG. Please note that WE follow the same process as FSM, to construct DAG. Moreover, the same approach is used to calculate *skipCost*. To skip elements of query, we do not had any links inside the DAG (it would be too complex to add between parent nodes at the given row to all the bottom rows). Instead, we directly modify the dissimilarity matrix for allowing the skips of query elements. We replace the cost of matching two elements (distance) by *skipCost* and we keep track of this jump to compute the warping path. To achieve that, we will have to modify the distance matrix. But first, to make the very first element of the query skip-able, one null element (0) is added at the beginning of both the query and target sequence (contrary to FSM), hence the size of the dissimilarity matrix changed into $\widehat{\mathfrak{D}}_{i,j}; 1 \leq i \leq p+1; 1 \leq j \leq q+1$. Then the modified dissimilarity matrix ($\mathfrak{Q}$) is created, by choosing the optimal option between matching with the amount of dissimilarity ($\widehat{\mathfrak{D}}_{i,j}$) and skipping with the burden of skip cost (*skipCost*) (see *Eq* 5.1). Another matrix ($\mathscr{M}_{i,j}$) is used to keep track on the indexes, where the condition: $skipCost < \widehat{\mathfrak{D}}_{i,j}$ is satisfied. Now, for calculating the path cost matrix $\mathscr{P}_{i,j}$, the $\mathfrak{Q}_{i,j}$ matrix is used instead of $\mathfrak{D}_{i,j}$ matrix. It is noteworthy to point out that mainly due to this useful modification over FSM architecture, ESC is able to skip noisy elements from query also.

[1]Please note that the indexes of all the matrix notations used in this chapter, are always starts from 1

Algorithm 8: EXEMPLARY SEQUENCE CARDINALITY

Input: $p, q, \mathfrak{Q}, \mathscr{S}(skipCost), \mathfrak{C}(smallSkipCost)$
Output: $\mathscr{P}_{p,q}, \mathscr{R}_{p,q}, \mathscr{C}_{p,q}$

$elasticity \leftarrow |q-p|$ (if not provided)
for $j \leftarrow 1$ ***to*** q **do**
  $\mathscr{P}(1,j) \leftarrow \mathfrak{Q}(1,j)$;
  ▷ Recording skip-able query elements (Eq. 5.1)
for $i \leftarrow 2$ ***to*** $p+1$ **do**
  for $j \leftarrow 1$ ***to*** $q+1$ **do**
    if $\mathfrak{Q}(i,j) > \mathscr{S}$ *&* $i > 1$ **then**
      $\mathfrak{Q}(i,j) \leftarrow \mathscr{S}$; $\mathscr{M}(i,j) \leftarrow -1$
    else
      $\mathscr{M}(i,j) \leftarrow 1$
for $i \leftarrow 2$ ***to*** $p+1$ **do**
  ▷ Deciding the parent nodes
  $L \leftarrow \max(1, [(i-1) - elasticity])$
  $R \leftarrow \min(q, [(i-1) + elasticity])$
  for $k \leftarrow L$ ***to*** R **do**
    ▷ Find right most child
    $D \leftarrow (k+1+elasticity) - max(0, |k-(i-1)|)$
    ▷ No many to one matching with 1^{st} dummy element
    if $i = 2$ *&* $k = 1$ **then**
      $C \leftarrow 2$
    else
      $C \leftarrow max(k, 2)$
    ▷ For each child node of parent node at column k
    for $j \leftarrow C$ ***to*** D **do**
      if $(j = k)$ **then**
        $\mathscr{J} \leftarrow \mathfrak{C}$
      else if $(j = k+1)$ **then**
        $\mathscr{J} \leftarrow 0$
      else
        $\mathscr{J} \leftarrow \mathscr{S} \times |j-(k+1)|$
      if $\mathscr{P}(i,j) > \mathscr{P}(i-1,k) + \mathscr{J} + \mathfrak{Q}(i,j)$ **then**
        $\mathscr{P}(i,j) \leftarrow \mathscr{P}(i-1,k) + \mathscr{J} + \mathfrak{Q}(i,j)$
        if $\mathscr{M}(i-1,k) = 1$ **then**
          ▷ No skipping of parent node
          $\mathscr{R}(i,j) \leftarrow i-1$; $\mathscr{C}(i,j) \leftarrow k$
        else if $\mathscr{M}(i-1,k) = -1$ **then**
          ▷ Skipping parent $(i-1,k)$
          $\mathscr{R}(i,j) \leftarrow \mathscr{R}(i-1,k)$; $\mathscr{C}(i,j) \leftarrow \mathscr{C}(i-1,k)$;
        ▷ For horizontal connection
      if $\mathscr{P}(i,j) > \mathscr{P}(i,j-1) + \mathfrak{C} + \mathfrak{Q}(i,j)$ **then**
        $\mathscr{P}(i,j) \leftarrow \mathscr{P}(i,j-1) + \mathfrak{C} + \mathfrak{Q}(i,j)$
        $\mathscr{R}(i,j) \leftarrow i$; $\mathscr{C}(i,j) \leftarrow j-1$

$$\mathfrak{Q}_{\mathrm{i,j}} = \begin{cases} \widehat{\mathfrak{D}}_{1,j} & 1 \leq j \leq q+1 \\ skipCost & \text{if } skipCost < \widehat{\mathfrak{D}}_{i,j} \\ \widehat{\mathfrak{D}}_{i,j} & \text{Otherwise} \end{cases} \tag{5.1}$$

The shortest path between each pair of reachable nodes in $\mathbb{G}$ can be found from path cost matrix $\mathscr{P}_{i,j}$. Considering $\mathscr{P}_{i,j}$ as a sub problem, the optimal structure of the problem can be formally defined as:

$$\mathscr{P}_{1,j} = \mathfrak{Q}_{1,j} \quad \text{if } 1 \leq j \leq q+1 \tag{5.2}$$

$$\mathscr{P}_{i,j} = \min \begin{pmatrix} \{\mathscr{P}_{i-1,k} + \mathfrak{Q}_{i,j} + \\ \qquad (skipCost \times (j - (k+1)))\} \\ \{\mathscr{P}_{i,j-1} + \mathfrak{C} + \mathfrak{Q}_{i,j}\} \\ \{\mathscr{P}_{i-1,j} + \mathfrak{C} + \mathfrak{Q}_{i,j}\} \end{pmatrix} \quad \text{if } \mathfrak{L}$$

$\mathfrak{L} : 2 \leq i \leq q+1; (i-1) - elasticity \leq k \leq (i-1) + elasticity; k+1 \leq j \leq (k+1) + elasticity - |k-(i-1)|$

This equation says that the 1^{st} row of distance matrix ($\mathfrak{Q}$) is copied to the path cost matrix ($\mathscr{P}$). Each cells that belongs to the i^{th} row is calculated by first choosing the possible parent nodes at previous row $(i-1)$ and at the column k ranging from $((i-1) - elasticity)$ to $((i-1)+elasticity)$. Then the connection with it's child nodes are performed in the same manner as FSM.

The distance between the two compared sequences can be obtained by getting the distance value, stored at the cell of path cost matrix, which represents the last match between the query and target elements (contrary to FSM). The back tracking process would start from the cell at the last row and at the j^{th} column of path cost matrix $\mathscr{P}$, where $p+1 \leq j \leq q+1$. To normalize the sequence dissimilarity value, we divide it by the number of corresponding pairs between the target and query sequence. Instead of having more facilities, Please note that, ESC has same complexity as FSM i.e. $\Theta((2.|q-p|^2).p)$. It is understandable from algorithm description that the main changes in ESC comes from the process of calculating values in path cost matrix (line 31 to 42), where we use some intelligent technique to skip outliers from query otherwise the complexity of iteration for path cost matrix calculation is same as FSM.

5.2.2 Description of the Proposed Pseudo Code

The cost matrix ($\mathscr{P}$) calculation process is given by Algorithm 8. All the cells of the matrix $\mathscr{P}$ are initialized with infinity. The cells of the path cost matrix ($\mathscr{P}$) are calculated (refer to lines $10-46$) by iterating i over each row where k iterates over parent nodes and j keeps track of child nodes. As an output, the algorithm provides the path costs in $\mathscr{P}$ along

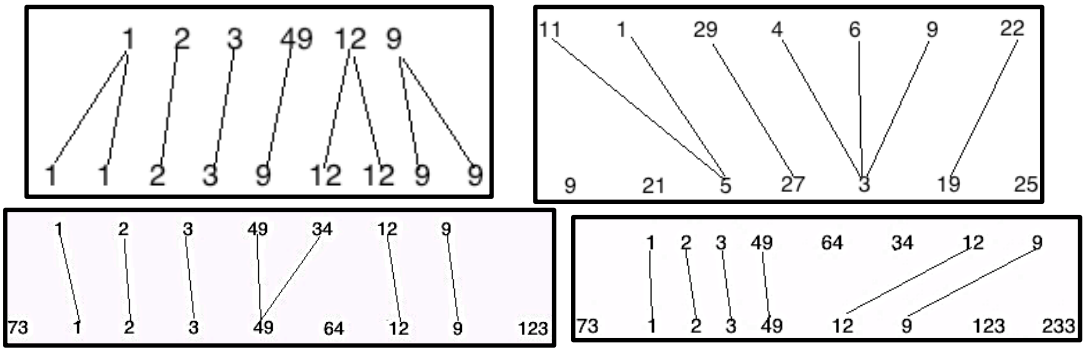


Figure 5.1: Matching ability of ESC on toy examples.

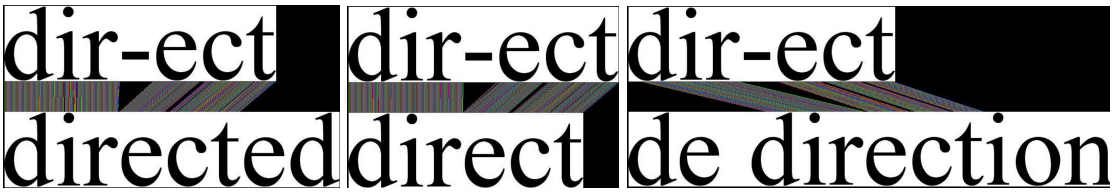


Figure 5.2: Matching ability of ESC on some artificial images.

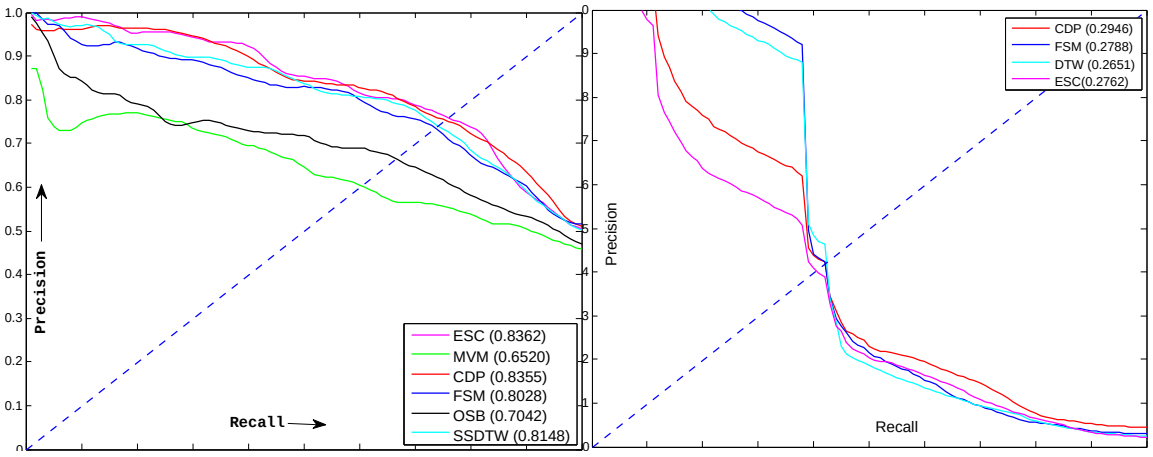
with two other matrices ($\mathscr{R}$ and $\mathscr{C}$) which are used to backtrack the shortest path, giving the closest correspondence between the query and target elements. The matrix $\mathscr{R}$ keeps track of rows and $\mathscr{C}$ keeps track of columns. For calculating each cell of a row, first the range of parent nodes are calculated (line 12-14). Next, for each parent node, the possible connections with it's child nodes are calculated from j which is iterated between C and D. In line 17, the system enforces to make diagonal connection with top left most parent node (the null elements added) with it's child node to avoid any many to one connection with this dummy node. Note that, full skip cost $\mathscr{J}$ is calculated by computing the number of skips $(j - (k + 1))$ in line 29. Using some toy examples, the behaviors of ESC and it's outliers skipping capability is demonstrated below in Fig. 5.1 and 5.2.

5.3 Experimental results on old manuscripts

To validate the interest of using ESC in word spotting application, experiments are performed on the CESR dataset. The used the same word spotting architecture, mentioned in Section 2.3 of Chapter 2. The segmented pseudo words or lines are described by a sequence of classical column based features mentioned in Section 2.3.2.1 of Chapter 2.

5.3.1 Results on CESR-Dataset-1

For performing the experiments with ESC and to check it's robustness, we comparatively choose small set from CESR dataset. We have manually selected 12 queries (based on number of occurrences, their possible relevant derivatives their meaning as queries). The test set, for each query, is composed of the pages of the book where the query or



(a) P-R plot of different sequence matching algorithms (mAP)

(b) Comparative performance of CDP, DTW, FSM and ESC

Figure 5.3: Comparative results of different sequence matching approaches on two separate datasets created from CESR data

its derivatives appear. The queries and their corresponding considered derivatives, are categorized below into 4 groups:

1. **cheualerie** (1), cheual (38), cheualier (4), **cheuallier** (1), **cheuaus** (4), **cheuaux** (8) : 124[2]
2. **cognoistre** (12), **cognoitre** (2), **connoistre** (12), connoitre (3), reconoitre (1), **recognoistre** (8), reconnoistre (7), reconnoitre (4): 128
3. **royaume** (2), royales (1), royau (2), roy (119), royaumes (4), royaus (1), royaute (2), royaux (1), royne (8), roys (17), rois (16): 282
4. **immortel** (14), immortales (1), **immortali** (10), immortalit (8), **mortel** (11), immortelle (10), immortels (2): 125

The aforementioned bold words are independently used as queries, whereas the other corresponding remaining words (bold and non bold) in the group are taken as the derivatives of considered query word. Please note that the aforementioned word derivatives for each query words are considered as true positives for calculating Precision-Recall (PR) of the system. The comparative precision-recall (PR) curve of CDP, FSM, SSDTW, MVM, OSB and ESC in Fig. 5.3a shows that ESC has minutely outperformed FSM and CDP algorithm. Especially, ESC has comparative high precision when recall is less than 0.5. The mean average precision is also mentioned in the graph, along with the curve of each algorithms. Nevertheless, even if ESC is efficiently working, we can notice that, on average, the improvement in matching is not able to highly influence the ranking order and precision and recall. One possible explanation comes from the features used, that are very local and

[2]Considered size of the dataset (for this particular group).

sensitive. In such situation, the ability of ESC to skip and jump elements could be counter productive if *jumpCost* are not correctly chosen (the algorithm becomes more sensitive).

$\breve{Q}^\circ$	$\widetilde{N}^\circ$	$\widetilde{O}^\circ$	$\bar{C}$	$\bar{F}$	$\bar{E}$
liberte	2	219	0.308	0.230	0.160
libre	19	2511	0.712	0.679	0.658
librement	3	462	0.363	0.374	0.165
TW = 3192; $\bar{AC}$ = 0.461; $\bar{AF}$ = 0.428 $\bar{AD}$ = 0.488; $\bar{AE}$ = 0.382					
mortel	11	1032	0.398	0.403	0.386
immortales	1	166	0.132	0.112	0.134
immortalit	8	968	0.332	0.290	0.343
immortel	14	930	0.200	0.181	0.209
immortelle	10	609	0.221	0.185	0.217
immortels	2	168	0.227	0.196	0.228
mort	28	2934	0.087	0.106	0.076
mortale	4	152	0.155	0.141	0.139
mortales	1	29	0.131	0.136	0.125
mortalia	5	275	0.107	0.074	0.103
mortalibus	2	285	0.078	0.044	0.073
mortalite	1	176	0.225	0.216	0.222
morte	10	1033	0.078	0.071	0.081
mortelle	9	822	0.285	0.290	0.286
mortelles	8	1172	0.264	0.228	0.269
mortels	4	324	0.227	0.225	0.270
mortem	5	243	0.036	0.026	0.030
mortes	1	169	0.054	0.101	0.046
morgue	2	217	0.026	0.024	0.027
morti	1	156	0.200	0.156	0.192
mortis	3	357	0.044	0.039	0.045
morts	3	221	0.082	0.099	0.066
mortua	1	184	0.031	0.025	0.032
mortuum	1	173	0.095	0.061	0.076
mortuus	1	184	0.030	0.024	0.030
mortz	3	539	0.076	0.092	0.071
TW = 13518; $\bar{AC}$ = 0.147; $\bar{AF}$ = 0.136 $\bar{AD}$ = 0.126; $\bar{AE}$ = 0.145					
reconneut	1	180	0.045	0.045	0.046
reconnoi	1	173	0.047	0.086	0.048
TW = 353; $\bar{AC}$ = 0.031 $\bar{AF}$ = 0.043 $\bar{AD}$ = 0.101 $\bar{AF}$ = 0.031					

$\breve{Q}^\circ$	$\widetilde{N}^\circ$	$\widetilde{O}^\circ$	$\bar{C}$	$\bar{F}$	$\bar{E}$
roy	119	12063	0.272	0.293	0.270
royales	1	154	0.066	0.059	0.066
royaume	2	263	0.646	0.765	0.640
royaumes	4	485	0.747	0.763	0.714
royaus	1	161	0.116	0.114	0.106
royaute	2	274	0.188	0.168	0.221
royaux	1	191	0.429	0.357	0.381
royne	8	915	0.252	0.193	0.231
roys	17	1442	0.173	0.158	0.171
rois	16	1894	0.021	0.040	0.022
roistre	1	110	0.012	0.013	0.013
roit	20	2744	0.017	0.015	0.015
TW = 20696; $\bar{AC}$ = 0.245;$\bar{AF}$ = 0.247 $\bar{AD}$ = 0.230; $\bar{AE}$ = 0.237					
despouille	4	355	0.721	0.701	0.669
despouiller	5	771	0.785	0.757	0.761
despouillera	1	152	0.763	0.768	0.807
depouille	1	181	0.754	0.719	0.736
depouiller	1	188	0.392	0.403	0.369
TW = 1647; $\bar{AC}$ = 0.683;$\bar{AF}$ = 0.670 $\bar{AD}$ = 0.602; $\bar{AE}$ = 0.669					
cheualier	4	522	0.501	0.470	0.534
cheual	38	5099	0.509	0.443	0.414
cheualerie	1	171	0.451	0.464	0.445
cheuallier	1	168	0.421	0.426	0.403
cheuaus	4	546	0.395	0.438	0.419
cheuaux	8	1064	0.400	0.334	0.366
TW = 7570; $\bar{AC}$ = 0.446; $\bar{AF}$ = 0.429 $\bar{AD}$ = 0.347; $\bar{AE}$ = 0.430					
victoire	23	3173	0.646	0.577	0.602
victo	1	125	0.593	0.511	0.397
victoi	2	325	0.542	0.368	0.383
victoire	23	3173	0.646	0.578	0.602
victorieus	5	523	0.375	0.337	0.274
victorieux	6	901	0.518	0.486	0.495
TW = 8220; $\bar{AC}$ = 0.553; $\bar{AF}$ = 0.476 $\bar{AD}$ = 0.497; $\bar{AF}$ = 0.459					

Table 5.2: Statistics and results of CESR dataset. The words, enclosed by rectangular box, represents each group.

All the notations are same as the ones mentioned in Table 4.5 of Chapter 4. $\bar{E}$ = mAP of ESC; $\bar{AD}$ = average mAP of DTW; $\bar{AE}$ = average mAP of ESC.

5.3.2 Results on CESR-Dataset-2

To verify the performance of ESC on comparatively larger dataset, we consider 60 queries from the set of 123 queries, mentioned in Section 4.4.2 of Chapter 4. The results on this set of queries are mentioned in the same manner (refer to Table 5.2) as the one is given in Table 4.5 of Chapter 4. It can be seen from the Table 5.2, that ESC has outperformed CDP and FSM in several cases (the highlighted entries in Table 5.2). The comparative P-R plot of ESC in comparison with CDP, DTW and FSM can be seen in Fig. 5.3. It can be seen from this plot that although ESC have not shown good performance at top ranked positions but at later ranked positions, it has performed well. The overall mAP (0.276) of ESC is also comparable with others; i.e. FSM (0.278) and DTW (0.265). Although, here also the overall statistical mAP of CDP has outperformed (0.294) others but it is shown in Table 5.2 that for some specific queries, ESC has outperformed all other techniques. This improvement is mainly due to it's noise skipping capability from query as well as target sequence.

5.4 Conclusion and Future Work

In this chapter, we presented a new robust sequence matching algorithm called as ESC algorithm. The ability to skip outlier elements present at any position of the target and **query sequence**, and the facility of **multiple matching**, makes the proposed ESC algorithm robust, generalized and applicable for various disciplines of sequence matching. In future, we plan to use hyphenated words as query and also try to retrieve composed hyphenated words. It is noteworthy to mention that if an empirical threshold can be fixed, which can distinguish between relevant and non relevant matches, ESC could also be used (e.g. CDP) for retrieving multiple occurrence of query sequence from long target sequence. But it needs more inspection, experiments and may be some minor modification of ESC's architecture. Moreover, like FSM, ESC can also be applied on conventional time series matching problems (UCR dataset). We plan to widely study the impact of the mentioned technique of calculating jump cost on various dataset. We also plan to perform more experiments to evaluate the robustness of ESC algorithm in comparison with other approaches on bigger and multilingual datasets.

Chapter 6

Improved Shape Code Based Word Matching For Indic Script

Contents

Abstract

In the previous chapters, we have seen several sequence matching techniques along with two new sequence matching technique i.e. FSM and ESC. All of these aforementioned techniques were build to match sequences of real numbers, where the numerical dissimilarity between numbers are considered for matching these sequences. For the case of matching two sets of symbols, *Longest Common Subsequence (LCSS)* and *Levenshtein Distance* are conventional techniques, which have been popularly used in literature. In this chapter, we propose a shape code (symbol) based word-image matching technique by using LCSS and *Levenshtein Distance* for word retrieval in documents, written in Indian languages. Each query word image to be searched is represented by a sequence of shape codes that corresponds to primitives. Then an inexact string matching technique is applied for measuring the similarity between the codes generated from the query word image and each candidate word images, obtained from the document. Based on the similarity score, we retrieve the document where the query image is found. Experimental results on Bangla, Devanagari scripts document image databases confirm the feasibility and efficiency of our proposed approach.

6.1 Introduction

There have been many attempt by the research community to propose several robust word spotting systems. The detailed literature review of numerous set of word spotting techniques are mentioned in Chapter 2. In this chapter, we only focused on the word spotting or word image retrieval techniques, which are achieved by shape coding based approaches. Word image retrieval can be considered as sub domain, descended from the domain of content based image retrieval (CBIR). In past few years, there have been many advanced technique proposed in this domain. There are numerous amount of applications of CBIR framework e.g. medicine, entertainment, education, manufacturing, etc. [Prasad et al., 2001], which make use of vast amount of visual data in the form of images. One of the main challenge in this domain is to retrieve images with rapidity along with satisfiable accuracy [Lew et al., 2006b]. The use of low level visual features such as color, shape, texture, spatial layout, object motion, etc., is a possible way to achieve this objective. One way to represent these features is by using shape codes, which can give sufficient distinguishable information in low dimensional space [Prasad et al., 2001].

Shape code is a technique to represent the image features by numeric values. As mentioned in earlier paragraph that shape coding based image matching techniques was initially introduced in the domain of image retrieval and have been well explored for word image retrieval, mainly for Latin scripts [Lu and Tan, 2008, Lew et al., 2006a, Lu and Tan, 2004]. Word Shape Coding based image matching technique is used for calculating the dissimilarity between query and target images. Word Shape Coding maps a word image into a set of numbers rather than real character identities. These symbol set can be thought as a sequence of real and positive numbers, extracted from query and target images. These sequence of real numbers can be matched by using several relevant sequence matching algorithms, for finding the dissimilarity values between query and target images.

Although there are sufficient amount of research work done for fast word image matching for Latin scripts [Lu and Tan, 2008], [Lew et al., 2006a], [Lu and Tan, 2004], but surprisingly there are few research works has been done in this direction with Indian scripts. More over, almost the same impediment do exist for Indian scripts that the OCR for Indic scripts are not mature and accurate enough for general purpose use. The goal of our research work is to propose a fast and robust word spotting technique for locating the desired words in the complete document, which is also useful for indexing purpose. In this chapter, a robust, script independent and computationally inexpensive word image matching process is proposed. This work is an extension of our previously published work [Tarafdar et al., 2010]. The chapter is organized as follows: the literature survey on shape code based word image matching techniques are explained in Section 6.2. The description and properties of Bangla and Devanagari scripts are given in Section 6.3. The outline of proposed approach, description of features and adapted sequence matching techniques are given in Section 6.4. The experimental results are presented in Section 6.5 and finally the conclusion of this chapter is mentioned in Section 6.6.

6.2 Literature Survey

There have been many earlier published works, which are often based on the character shape coding that annotates character images by a set of pre-defined codes, for the retrieval of document images. In [Shijian Lu, Chew Lim Tan, 2006], the character extreme points are used to code segmented word images and the resultant word shape codes are then compared by using *Bray Curtis* distance [Shijian Lu, Chew Lim Tan, 2006] for the language identification. Later, along with the features given in [Shijian Lu, Chew Lim Tan, 2006], the number of horizontal word cuts is incorporated in [Lu and Tan, 2008] for the multilingual document image retrieval. In this case the authors used cosine distance as the dissimilarity measure. Besides this approach, another similar approach is also reported for keyword spotting in [Lu and Tan, 2004], which explains the capability of partial word matching. In this approach, each word image is firstly annotated by a primitive string. Then, an inexact string matching technique is utilized to measure the similarity between the two primitive strings generated from two word images. On the basis of previous works mentioned in [Lu and Tan, 2008], [Lew et al., 2006a], [Lu and Tan, 2004] an alternative technique and combination of feature descriptors for keyword spotting is mentioned in [Bai et al., 2009]. The different sequence alignment similarity measures, mentioned in this paper is used for partial or whole word matching. The mentioned technique is tolerant to serifs, font styles and certain degrees of touching, broken or overlapping characters. Compared to the previous works by the same authors mentioned in [Lu and Tan, 2008, Lu and Tan, 2004], this approach shows some improvements not only in terms of better precision and higher retrieval rate, but more importantly, the ability for partial matching also.

A technique to locate content representing words for a given document image using abstract representation of character shapes is described in [Nakayama, 1994]. The ambiguity of mapping of character shape code is limited by distinguishing punctuation marks from general characters and stop words from other words, based on the permutation of character shape codes. Numerals and acronyms in capital letters are also distinguished from general words by the arrangement of the shape codes. With these distinguishable characteristics, the content representing words in a document are marked. Another word image matching technique, based on shape codes for hand-written English document images is mentioned in [Sarkar, 2013]. This technique implements two level of selection for word segments to match search query. First based on word size and then based on character shape code of query.

All of the above mentioned techniques are based on character shape codes, so obviously they need properly segmented characters. But this a critical bottleneck of these systems. Even on minutely bad quality images, it could be difficult to properly segment characters. In such cases, these above defined systems would fail to perform. To overcome the problems of character shape coding scheme, several word shape coding schemes has been proposed in the literature, which treats each word image as a single component and so are much tolerant to character segmentation error. In [Shijian and Lim Tan, 2008], vertical bar pattern based features are extracted from segmented word images through local extrema point detection. These extracted vertical bar pattern are used to build document vectors, which is used to characterize the shape and frequency of the contained character and word images. It in terms of help to obtain the pair-wise similarity of document images by means

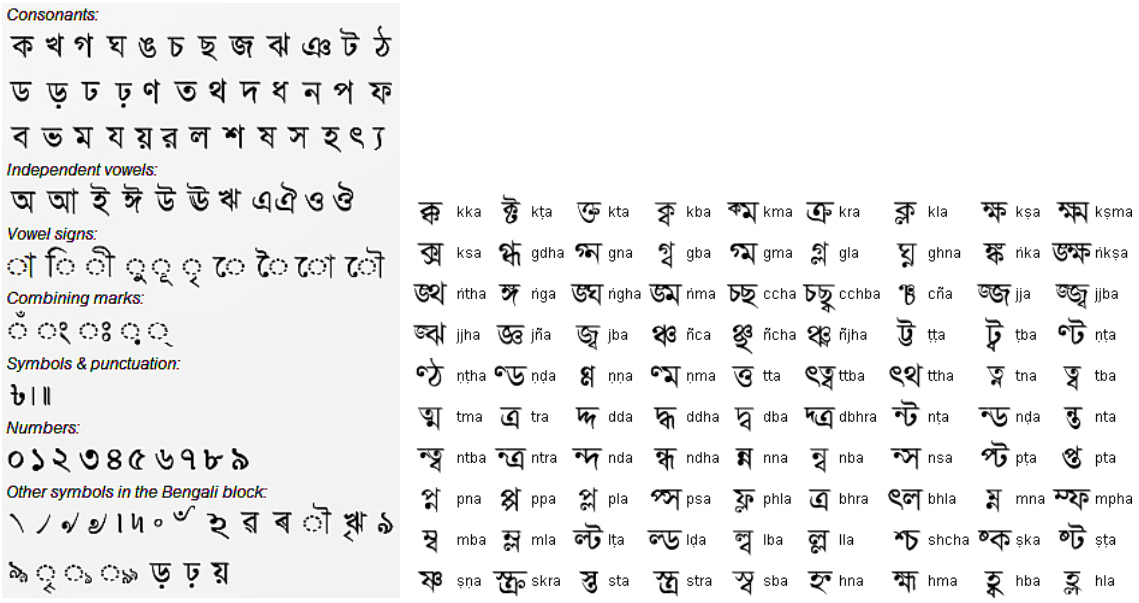
of scalar product of document vectors. Based on the density and distribution of vertical runs, underlying script of the document is identified. Latin based languages are then differentiated by using a set of word shape code, which are constructed using horizontal word runs and character extreme points. Another word shape token based technique for document image classification is explained in [Nakayama, 1996]. Using a vector space classifier with scanned document image database, this technique has shown that word shape token based approach is comparatively accurate for content oriented categorization compared to conventional OCR based approach.

A text retrieval system is proposed in [Tan et al., 2002]. In this approach, documents are segmented into character objects and image features are extracted, namely the vertical traverse density (VTD) and horizontal traverse density (HTD). An *n-gram*-based document vector is constructed for each document based on these features. Text similarity between documents is then measured by calculating the dot product of the document vectors. A technique is mentioned in [Smeaton and Spitz, 1997] for performing information retrieval on document images by the generalizations of character images, which are implicitly used for classification. The shape codes, extracted from character images agglomerates into word tokens, which is used as a representation of the underlying words for word retrieval purpose. A system is explained in [Chen et al., 1995] for searching user-specified phrases in imaged text. The phrases can be word fragments, words, or groups of words. The imaged text can be consisting of a number of different fonts and can also contain graphics. A combination of morphology, simple statistical and hidden Markov model are used to detect and locate the phrases on the deskewed images based on multi-resolution morphology. Baselines, top lines and the x-height in a text-line are identified using simple statistical methods. Hidden Markov models are created for each user-specified search string and to represent all text and graphics other than the search strings. Phrases are identified using Viterbi decoding on a spotting paradigm created from these models. A method is described in [Spitz, 1997] is to classify the scripts of a document into two broad classes: Han-based and Latin-based. The approach of classification is based on the spatial relationships of features related to the upward concavities in the existing character structures. Language identification within the Han script class (Chinese, Japanese, Korean) is performed by analysis of the distribution of optical density in the text images. Whereas 23 Latin-based languages detection technique based on character shape codes, a representation of Latin text that is inexpensive to compute is also explained in this paper.

6.3 Properties of Bangla and Devanagari Scripts

There are twelve scripts in India and in most of these scripts, the number of alphabets (basic and compound characters) is more than two hundred fifty. Bangla and Devanagari [Chaudhuri and Pal, 1997,Chaudhuri and Pal, 1998] are oriental scripts descended from the Brahmi script and these scripts are the two most popular scripts in India. In both scripts, the writing style is from left to right and there are no concept of upper/lower case. Both the scripts have about fifty basic characters. The basic characters of Bangla script is shown in Fig.6.1. The most popular Indian language Hindi, which is written in Devanagari script.

[1]Source: `http://rishida.net/blog/?cat=3`; `http://www.omniglot.com/writing/bengali.htm`

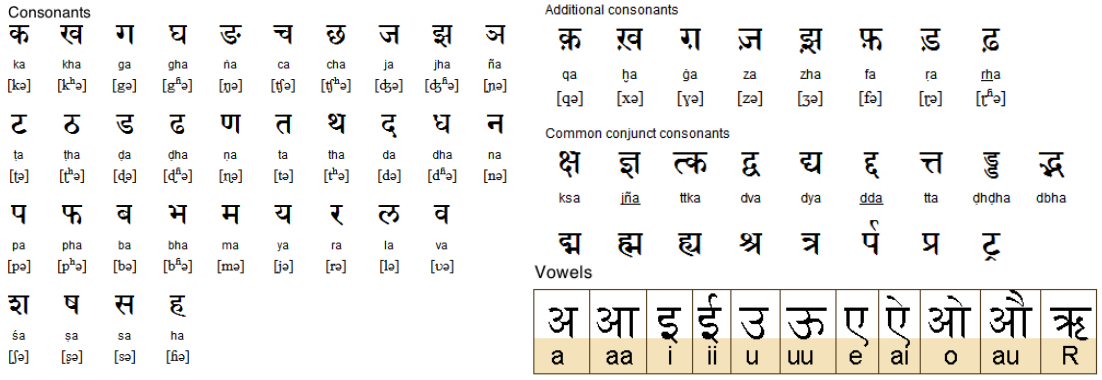


(a) (b)

Figure 6.1: (a) Basic characters and (b) compound characters of Bangla script[1]

Nepali, Sanskrit and Marathi are also written in Devanagari script. Moreover, Hindi is the national language of India and the third most popular language in the world [Lin-Lin, 2009]. Devanagari script has 52 symbols (10 vowels, 2 modifiers and 40 consonants) (see Fig. 6.2a and Fig. 6.2b). Alphabets are known as "matra" symbols. Matra symbols are used when consonants and vowels are to be written together. Bangla, the second most popular language in India and the fifth most popular language in the world, is an ancient Indo-Aryans language. Bangla script alphabet is used in texts of Bangla, Assamese and Manipuri languages. Bangla is also the national language of Bangladesh and the official language of West Bengal State of India. Bangla is spoken by around 211 million people in Bangladesh and Indian state of West Bengal. Alphabet set of this script has 11 vowels, 40 consonants, and 10 numerals called basic characters (see Fig. 6.1a). There are also more than 200 compound characters which is formed by combination of two or more basic characters. Some examples of compound characters are shown in Fig.6.1b. Vowels in these scripts generally take a modified shape in most words and are called modifiers or allographs (see Fig. 6.1b and Fig.6.2b). Modifiers generally do not disturb the shape of basic characters in the middle zone of a line. If the shape is disturbed in the middle zone, we call resultant shape as a compound character. A segmented word of such scripts can be partitioned into three zones. The upper zone denotes the portion above the headline, the middle zone denotes the portion between headline and baseline and the lower zone is the portion below baseline. The imaginary line separating middle and lower zone is known as base line. Different zones of a Devanagari line are shown in Fig.6.3a. Horizontal histogram technique is used, to find the headline and baseline [Chaudhuri and Pal, 1998] of a word

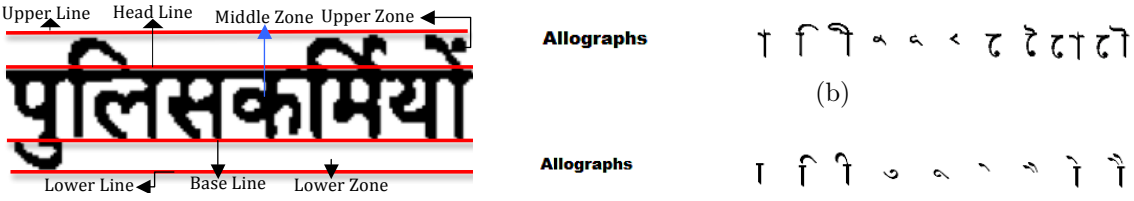
[2]Source: http://www.omniglot.com/writing/hindi.htm;
http://www.airliners.net/aviation-forums/non_aviation/print.main?id=2263974



(a) (b)

Figure 6.2: (a) Basic consonants of Devanagari script (left) (b) Additional consonants, compound consonants and vowels of Devanagari scripts (right)[2]

image.



(a) (b) (c)

Figure 6.3: (a) Different zone of the word (b) Some Bangla (top)[3], (c) Devanagari (bottom) characters[3]

6.4 Outline of the proposed approach

For this work, we are considering improperly segmented words, which are directly available from the experimental dataset[3] (details of dataset is given in Section 6.5). By considering the available monochrome segmented words for our experiments, the features are extracted from each word image and are stored off-line. The purpose of word spotting is then achieved by calculating the dissimilarity measures between query and target word images (i.e. all the word images belongs to the dataset except query words). After calculating the dissimilarity values between target word images, these images are ranked according to their dissimilarity values. The details of the different steps are given below.

[3]We did not have access of complete document images, we only had access to the segmented words. By visually inspecting the dataset, it can be understandable that the applied word segmentation algorithm could not perfectly segment all the words and there are some occurrences of over and under segmentations.

6.4.1 Feature Extraction

In the following section, we briefly elucidate five word shape coding schemes. Interested readers are requested to see [Tarafdar et al., 2010], for detailed discussions on these features. For general discussion on shape codes, used for word image matching (mainly on Latin scripts), interested readers are requested to see [Lu et al., 2008, Tan et al., 2003]. In this research work, we use all the features, mentioned in [Tarafdar et al., 2010]. Moreover, we propose three new features based on the fore ground pixels ($F6, F7, F8$; for details, please see Section 6.4.1.6). The effectiveness of these added features are shown in experimental section (refer to Section 6.5). In the following sections, we present a brief descriptions of the features, taken from [Tarafdar et al., 2010] and at last but not least, we present the description of the proposed features ($\bar{F}6, \bar{F}7, \bar{F}8$).

Let's denote the considered binarized word image by W_b having total R number of rows and C number of columns. Since in Devanagari, Bangla and Gurumukhi text, characters in a word are connected through headlines, we detect and delete the headline portion of the word, to make the characters of a word *"more disconnected"* as shown in Fig.6.4i. To remove the headline, a horizontal histogram of foreground pixels are calculated. So, obviously the rows belongs to headline will participate for generating highest histogram peak. Such rows are identified and removed. Later the upper and lower separated parts are merged together (see Fig.6.4i) for maintaining right form of the word. This operation of headline removal generates disconnected characters (see Fig.6.4i). After obtaining the disconnected characters, the middle zone of the word is detected by removing the head line and by locating the base line of a word. The head line and base line are detected by simple horizontal projection of fore ground pixels (see Fig.6.3a). Please note that, the features $F1 - F5$ (details are given after) are calculated from middle zone of the word.

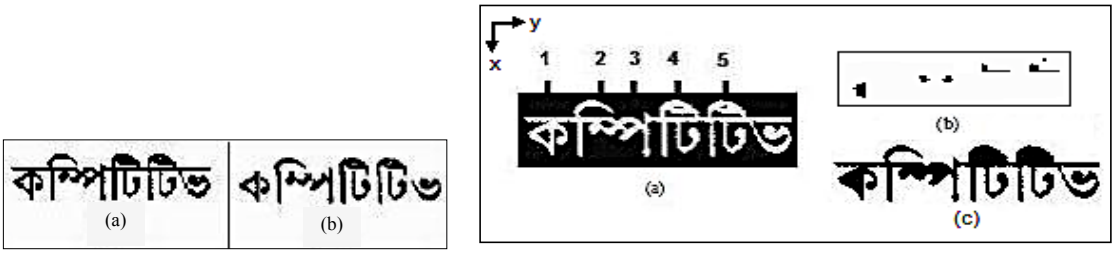
6.4.1.1 Coding based on loop position

Number of loops and their positions in the word image is a distinguishable property of each word. To use this information as a feature, we find the center of gravity (C.G.) of each existing loops in the image. Then, we normalize the x coordinate of the C.G.'s of these loops ($C.G._x$ in Eq^n 6.1), with respect to the width of the word. The following Equation 6.1 is used for calculating this feature value. To find the loops, the word image pixels are initially inverted (foreground pixels changed into background pixels and background pixels are changed into foreground pixels) and then connected components are extracted from the inverted image. Please note that, for avoiding the cases where foreground pixels are connected with the bounding rectangle of the word image, we perform padding of one background pixel along the perimeter of the image. Among the connected components, the biggest one is the background component, so it is ignored, but other connected components are considered as loops.

$$F_i^1 = 10 \times (C.G._{i_x}/C); 1 \leq i \leq N_b \tag{6.1}$$

Where N_b is the number of loops and $C.G._{i_x}$ corresponds to x positions of C.G. of the

loops in the word relative to image width C in pixels.



(i) (a) Original image. (b) Headline removed image

(ii) (a) Inverted Image[3] (b) Extracted loops[3] (c) Fill up loops are shown[3]

Figure 6.4: (i)The original and corresponding headline removed image. (ii) Techniques to obtain loops in the image.

For example, the five loops in Fig.6.4ii-(a) are shown in Fig.6.4ii-(b), positions of five loops are marked by 1, 2, 3, 4 and 5 in Fig 6.4ii-(a). The x coordinate of the C.G. for each of the five loops, computed from left to right, are 11, 43, 55, 79 and 104 respectively, and the width of the word is 140. Using these x-coordinates in the above formula and dividing it by the word width (140) we get the shape code as 03357.

6.4.1.2 Coding based on background components

Background can provide useful information that helps in word spotting scheme. Run length information of the background region between two consecutive characters of the middle zone portion is used for this purpose. The background between two characters is initially detected. Such background portions are marked in black in Fig. 6.5. Here, we have ten components but we do not consider the leftmost and the rightmost components. We also

Figure 6.5: Inverted busy zone portion of Fig.6.4i-(a)[3].

do not consider the loops for background components. Thus the word shown in Fig.6.4i has 8 background components. After obtaining background components, horizontal scanning is applied on each background component to calculate the maximum ($\mathfrak{Max}$) and minimum ($\mathfrak{Min}$) horizontal runs among all the horizontal runs in each background components. For each background component, the shape code feature is obtained by calculating two separate code values, mentioned in Equation 6.2 and the final code is given by Equation 6.3.

[3]Figures are used in this thesis, with the written permission from [Arundhati Tarafdar, Ranju Mondal, Srikanta Pal, Umapada Pal and Fumitaka Kimura, "Shape Code based Word-image Matching for Retrieval of Indian Multi-lingual Documents", ICPR, 2010. © 2010 IEEE.]

$$\bar{F_i^2} = 10 \times \mathfrak{Max}_i/(\zeta + 1)$$
$$\hat{F_i^2} = 10 \times \mathfrak{Min}_i/(\zeta + 1) \quad (6.2) \qquad F^2 = \bar{F_i^2} \frown \hat{F_i^2} (\text{concatenating } \bar{F_i^2} \text{ and } \hat{F_i^2}) \quad (6.3)$$
$$1 \leq i \leq N_b$$
$$\zeta = max(\mathfrak{Max}_i)$$

Feature values are normalized through dividing by $(\tau + 1)$, where τ represents longest horizontal run among all $\mathfrak{Max}_i$ values, found in a word image. For each background component, we get two code values ($\bar{F_i^2}$ and $\hat{F_i^2}$). So, if a word have N_b numbers of background components then the length of the code string of the word would be $2 \times N_b$.

6.4.1.3 Coding based on extreme points

To obtain this feature, each word is horizontally segmented into two parts through the middle row between head line and base line. Now, from each disconnected component of the upper and lower part, the locations of extreme points i.e. upper and lower extreme points are calculated respectively and these extreme points are considered as feature values (F_3). The extreme points of upper and lower parts of Fig.6.4i-(b) are marked by gray dot in Fig.6.6-(a) and 6.6-(b) respectively. Now six coding values are employed according to

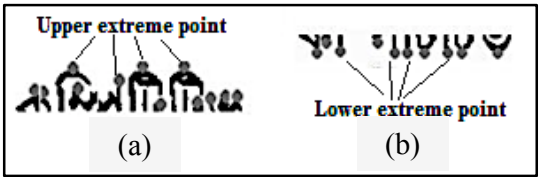


Figure 6.6: Local extreme points of (a) upper and (b) lower portions of a word shown in Fig.6.4i-(a). Extreme points are marked by gray dot[3].

the position of extreme points. We use code 1 if an extreme point lies above headline, code 2 if extreme point lies on headline portion, code 3 and 4 respectively, if extreme point lies in upper and lower half of middle zone, code 5 if extreme point lies on baseline and finally code 6 if extreme point lies in lower zone. For example, the code obtained from the Bangla word shown in Fig.6.4i-(a) is 5234512415515351533353. Here components are used from left to right to get positional information for coding. Since there are 22 extreme points in Fig.6.6 (12 upper and 10 lower extreme points), so, the length of this coded string is 22.

6.4.1.4 Vertical shape based coding

In this coding scheme, the column positions (c) of the vertical lines like structure in a word are located. Based on these positional values, the normalized coding is obtained by using the following formula.

$$F_4 = (10 \times c)/C;\ C = \text{Width of the image} \tag{6.4}$$

For example, the coding value of the word shown in Fig.6.4i-(a) is 1245577. Since the word shown in Fig.6.4i-(a) has 7 vertical lines. The vertical line like structures are obtained by finding the columns having long run of foreground pixels (black run). If the length of black run is greater than 65% of the height of middle zone, such columns are considered as vertical line like structures. If we get multiple vertical lines in consecutive columns, then we will consider the last column among them and ignore the remaining ones (vertical bar, having more than one pixel thickness).

6.4.1.5 Crossing based coding

The portion of the image between two consecutive vertical lines are divided into four equal parts by dividing the intermediate region through three column positions (see Fig. 6.7). After that, the number of transitions from foreground to background and/or background to foreground in each of these three columns are counted. The coding (F_5) is done based on these crossing values.

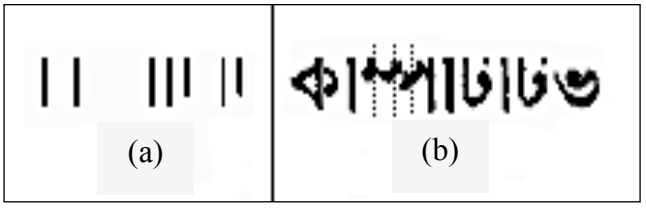


Figure 6.7: (a) Extracted vertical lines from the busy zone of Fig.6.4i-(a)[3]. (b) Example of division of distance between two consecutive vertical lines into four equal parts[3].

6.4.1.6 Coding based on C.G of foreground pixels

To calculate this feature, initially the image is divided into slits, based on the amount of foreground pixels it contains. The full image is divided in such a manner, such that each slit will contain τ amount of F.G pixels (τ is heuristically set to 100). So the total number of slits, are not same and it depends on the number of foreground pixels. In general, longer words will have more F.G pixels, hence it would have more number of slits and shorter words will have less number of slits.

Three features (F_6, F_7, F_8) are calculated from each slit, based on the position of center of gravity (C.G), obtained from the locations of foreground pixels in the slit. As shown in Fig.6.8, each slit (red rectangle) is divided into regions in 2 different ways. Firstly, the slit is divided into 8 equal sized blocks. The horizontal (F_6) and vertical (F_7) shape code for each block is mentioned in the image (left). This shape codes are nothing but the coordinates of the blocks, e.g. the vertical and horizontal shape code of the middle block is 2 and 2 respectively. Secondly, the slit portion is divided into three regions, as

shown in Fig.6.8 (right one). The shape code (F_8) is given by the region number, where the C.G. of foreground pixels belongs to (the code for each region is mentioned in the Fig.6.8). As visible in the Fig. 6.8 that the image is divided into even number of small blocks along horizontal and vertical direction. Based on these blocks, the three regions marked in different colors are identified.

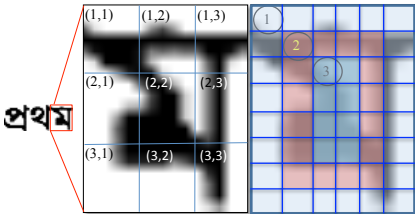


Figure 6.8: C.G of foreground pixels based feature

6.4.2 Sequence Matching Techniques

After calculating the feature values and representing them in feature vectors, sequence matching techniques are applied for performing the sequence matching between query and target feature vectors. For matching query and target images, we applied basic level of pruning by only considering those target words, whose width is at least greater than or equal to the half of query's width. Please note that, as we are also looking for partial word matching, longer words are considered for matching. Based on the different features described above, each candidate word image is described by numeric codes. The word-searching problem can then be stated as finding a particular sequence/subsequence in the codes of the candidate and query words. The procedure of matching word images then become a measurement of the similarity between the code $X = x_1; ...; x_m$, representing the features of the query word and the code $Y = y_1; ...; y_n$, representing the features of a candidate word image. The process of finding similarity between two sequence X and Y is described in the following sub section. We propose an combined matching approach for calculating distance between the query and target sequence. This combined approach is fabricated by sequentially linking two individual matching techniques : *Longest common sub-sequence* and *Levenshtein Distance*. Use of LCSS helps to find the longest common subsequence between two feature vectors, whereas *Levenshtein Distance* calculation helps us to get total number of required edit operations for making one common sub-sequence equal to another. This shape code based matching process is quite fast, compared to other image matching techniques [Manmatha, R. Chengfeng and Riseman, 1996], [Khurshid et al., 2012]. Thanks to the low dimensional features of shape code, LCSS is able to find quickly the common subsequence, contrary to high dimensional column based features (refer to Section 2.3.2.1 of Chapter 2). For detailed discussion on LCSS, please see Section 3.7.1 Chapter 3. The reason for choosing LCSS is that, in the presence of noise (we have some noisy data), theoretically LCSS should show more accurate matching than DTW. It is noteworthy to point out here that in LCSS, we use a matching threshold ($0 < \sigma < 1$) to compare two points, coming from two sequences. These two points are considered as same if their distance is less than σ. For our case, we considered $\sigma = 0$, i.e. we are looking for

perfect match between two elements, in other words, we consider two codes are same if and only if they are equal. After obtaining the longest common subsequence between two feature vectors, we calculate *Levenshtein Distance* between these common subsequences.

6.4.2.1 Levenshtein Measure

This measure is defined as the minimum number of operations needed to transform one string into other. Here an operation is defined as an insertion, deletion, or substitution of a single code, or a transposition of two codes using Damerau Levenshtein [4] distance value.

$$\mathfrak{S}(i,j) = \begin{cases} j & if \;\; i = 1; j > 1 \\ i & if \;\; i > 1; j = 1 \\ \min \begin{cases} \mathfrak{S}(i-1,j)+1 & \\ \mathfrak{S}(i,j-1)+1 & if \;\; i > 1; j > 1 \\ \mathfrak{S}(i-1,j-1)+Cost & \end{cases} \end{cases} \tag{6.5}$$

$$Cost = \begin{cases} 0 & if \;\; X_i = Y_j \\ 1 & if \;\; otherwise \end{cases} \tag{6.6}$$

The Levenshtein distance between two sequence X and Y is obtained by the value at $\mathfrak{S}(m,n)(Equation\ 6.5)$; where $m = n =$ length of the common sub-sequence.

6.4.2.2 Adapted Matching Strategy

The features extracted from one query and one target image can be represented as: $Q^F = \{g^1_{1,...p}, ..., g^8_{1,...q}\}$ and $T^F = \{g^1_{1,...p'}, g^8_{1,...q'}\}; \{p, p', q, q'\} \in \mathbb{R}^+$. The length of each feature vectors i.e. $f^1..f^8$ are not same for a particular image and it could be different from one image to another also. It may also occur that some features does not exists for some particular images. For a particular query Q_i, let's say that it has t ($t \in \{1..8\}$) number of existing features. Now, with reference to the query image, we choose all the target images, which has k ($k \geq t-1$) numbers of existing features. After choosing the target images and their relevant features, LCSS is applied on the corresponding features, extracted from query and target images respectively, e.g we apply LCSS on g^1_Q of query and g^1_T of target i.e. $LCSS(g^1_Q, g^1_T)....LCSS(g^k_Q, g^k_T)$ etc. LCSS will give the longest common sub-sequence between the feature vectors. Hence, from the returned indexes of longest common subsequences, we can obtain the 1^{st} and last query and target element's indexes, which are common between both the feature vectors. Now the next task is to calculate Levenshtein distance between these parts of query and target sequence. Levenshtein distance measure is able to give the number of edit operation to be performed, for making two comparable sequence equivalent. So, from each comparison of feature vectors, we can obtain the length of longest common sub-sequence and it's corresponding edit distance. The total length of

[4] `http://www.levenshtein.net/index.html`

LCSS and total edit distance, for a particular target image is calculated by adding up all the lengths of longest common sub-sequences and distances, obtained from each feature vectors of a target image. In the Fig.6.10, only one feature vector matching is shown but as explained before, LCSS is applied to all the feature vectors. For example, when LCSS is applied to the following shown feature vector in Fig.6.10, the obtained result is shown in Fig.6.9. After getting the first and last index of common sub-sequence, we calculate edit distance between all the elements from target and query, whose index belongs in between the first and last indexes of common sub-sequence, i.e. in this case, we calculate Levenshtein distance between $v_1 = \{1, 9, 2, 9, 4\}$ and $v_2 = \{1, 1, 4\}$

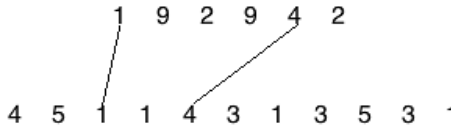


Figure 6.9: Matching of elements by LCSS

Now for giving overall ranking of the nearest matches, a parametric weighting based technique is applied to use the participation from both the measures, obtained from total longest common sub-sequence and edit distance. If the ranking is only done by using total longest common sub-sequence in descending order (higher the total length, closer the match), then the obtained mean average precision (mAP) of the system is **0.557** and if only the total edit distances is ranked in ascending order (lesser the total distance, closer the match), mAP of the system becomes **0.241**. From these result, we can observe that ranking based on length of longest common sub-sequence is better than the one based on edit distance. Although statistically the overall mAP, only by edit distance is not better but we observe that for some candidates the edit distance works better than other one (length of common subsequence). So, the following formula is used to obtain the combined distance, by weighting individual participation of total distance and common sub-sequence respectively.

$$\mathfrak{D}_{combo} = \alpha \times \mathfrak{D}(Q^F, T^F) + \{(1 - \alpha) \times \{\mathscr{L}_{max} - \mathscr{L}(Q^F, T^F)\}\} \tag{6.7}$$

In the above where Equation 6.7, $\mathfrak{D}(Q^F, T^F)$ represents the total Levenshtein distance between query (Q^F) and target (T^F) feature vectors. The notation $\mathscr{L}(Q^F, T^F)$ represents the total length of common sub-sequences of query and target feature vectors (remember that query and target can have 8 pairs of separate feature vectors), whereas $\mathscr{L}_{max}$ represents the maximum of such lengths. The reason of subtraction in the last term is to sort the distance in descending order instead of ascending order (in contrast with usual way of sorting the distance in ascending order). Now, when the combined distance ($\mathfrak{D}_{combo}$) is calculated, those distances are sorted in descending order and the mAP is calculated. The value of α is set from the learning data set. Details of setting α is explained in the experimental section. The Fig.6.10 shows the process of matching. The left container (pinkish one) represents all the extracted feature vector from each of the target images (N). The extracted feature vectors from the query image are shown in the right container (blueish one). Each feature vector of query is matched with the corresponding feature vector of target by LCSS algorithm.

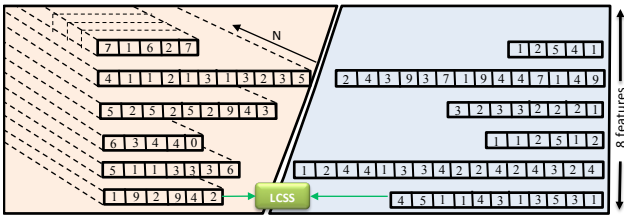


Figure 6.10: Feature level comparison by LCSS

6.5 Experimental Results

The proposed word spotting method is tested comprehensively on two types of documents. For this experiment, variety of printed documents (e.g. books, newspaper, magazines etc) of Bangla, Devanagari scripts are considered. The above shown Table. 6.1 shows the statistics of Bangla and Hindi dataset, used for this experiment.

Dataset	$\widetilde{N}$	$\hat{N}$
Bangla	14	3234
Hindi	15	4860

$\widetilde{N}$ = No. of query images
$\hat{N}$ = No. of target images

Table 6.1: Statistics of the used dataset

A general experimental protocol of word spotting is followed here. Distance between the query and target images are calculated and according to their dissimilarity values, the target images are ranked. Mean average precision (mAP)[5] is chosen to evaluate the performance of the system. Initially, we heuristically set the value of $\alpha = 0.5$ and calculate mAP by determining the combined distance value from Equation 6.7. To verify the effectiveness of the features $\bar{F}_1 - \bar{F}_5$, we experimented the evaluation setup on original *Bangla* dataset and we obtained **mAP = 0.6831**. Now to check the effect of the added features $(\bar{F}_6 - \bar{F}_8)$, we used all the features together (i.e. $\bar{F}_1 - \bar{F}_8$) and we obtained **mAP = 0.6935**. So from these accuracy values, it can be concluded that the added features $(\bar{F}_6 - \bar{F}_8)$ have a little positive impact but it is less computationally complex. The feature $\bar{F}_6 - \bar{F}_8$ takes 0.0035 sec. to compute, whereas other features i.e. $\bar{F}_1 - \bar{F}_5$ takes 0.0187 sec.[6] to compute in Intel Xeon CPU with 8 GB RAM [7]. To check the robustness of the proposed technique, we apply artificial Gaussian noise of difference intensities in the query and target images respectively. The effect of noise can be visible in Fig.6.11. It is visible from Table 6.2. that the effect of noise can highly affect the accuracy of the system. As shown from the Table 6.2. $F_1 - F_8$ are more robust compared to $\bar{F}_1 - \bar{F}_5$

As mentioned earlier, the value of α was chosen heuristically for performing all the above mentioned experiments. But for the proper and unbiased selection of α, we use cross validation technique. From each dataset, two queries are chosen randomly and the best

[5] http://en.wikipedia.org/wiki/Information retrieval#Mean average precision

[6] Averaged on 10 word images.

[7] Please note that, our implementation in MatLab is unoptimized. The computational speed of the system can be further increased by optimizing the code.

Table 6.2: Accuracy in different level of added noise

	σ	**0.0**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
mAP	F1-F8	0.691	0.691	0.691	0.691	0.691	0.678	0.506	0.429	0.187	0.165
	F1-F5	0.683	0.683	0.683	0.683	0.681	0.645	0.537	0.371	0.154	0.122

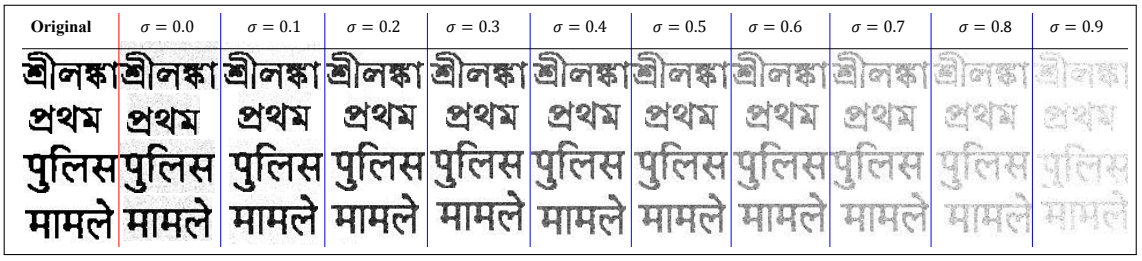


Figure 6.11: Effect of noise in 2 Bangla and 2 Hindi words. The left most image in each row is the original image whereas other noisy images (noise intensity: $\sigma = 0 - 0.9$) are shown in respective columns (figures are best visible in digital copy).

value of α is searched. After obtaining this best value of α on two queries, this value is used for all other remaining queries. This process is repeated 10 times and the average mAP (A-mAP) is given as the final result of the system. When this setup of experimented for Bangla and Hindi dataset respectively, we obtained A-mAP value as **0.7112** and **0.6464** respectively. When the same setup is experimented by only using feature set $\bar{F}_1 - \bar{F}_5$ on Bangla and Hindi dataset respectively, we obtain A-mAP value of **0.6938** and **0.6358**. So, again it can be visible that the added feature $\bar{F}_6 - \bar{F}_8$ has helped to increase the accuracy of the system.

6.6 Conclusion

This chapter presents a fast shape code based word matching technique, where the dissimilarity between two words are measured by the parametric combination of LCSS and Levenshtein distance. A strong advantage of this system is it's speed whereas the robustness is only tested on two Indian languages. To verify the robustness of the proposed method, we would like to test it for other scripts also. There are some features, used in this chapter (e.g. loops, vertical shape, back-ground),that would not be suited in other scripts. More research work is needed to explore some robust and script independent set of features. In future, we would like to adapt the proposed technique for spotting words in segmented lines, instead of segmented words.

Publications

Journals

T. Mondal, N. Ragot, J.-Y. Ramel, and U. Pal, **"Flexible Sequence Matching Technique : An Effective Learning-free Approach For Word Spotting"**, submitted to *Pattern Recognition*, Elsevier.

International Conferences

T. Mondal, N. Ragot, J.-Y. Ramel, and U. Pal, **"A Fast Word Retrieval Technique Based on Kernelized Locality Sensitive Hashing"**, 2013 12th International Conference on Document Analysis and Recognition, pp. 1195-1199, Aug. 2013.

T. Mondal, N. Ragot, J.-y. Ramel, and U. Pal, **"Flexible Sequence Matching Technique: Application to Word Spotting in Degraded Documents"**, in ICFHR. IEEE, Sep. 2014, pp. 210-215.

T. Mondal, N. Ragot, J.-y. Ramel, and U. Pal, **"Exemplary Sequence Cardinality : An Effective Application for Word Spotting"**, in ICDAR. IEEE, Aug. 2015.

T. Mondal, N. Ragot, J.-y. Ramel, and U. Pal, **"Performance Evaluation of DTW and its Variants for Word Spotting in Degraded Documents"**, in ICDAR. IEEE, Aug. 2015.

T. Mondal, A. Tarafdar, N. Ragot, J.-y. Ramel, and U. Pal, **"Improved Shape Code Based Word Matching For Multi-script Documents"**, to be appeared in Proceedings of the 3rd Asian Conference on Pattern Recognition, Nov. 2015.

T. Mondal, N. Ragot, J.-y. Ramel, and U. Pal, **"Constrained and Parametric Dynamic Programming for Word Image retrieval"**, Submitted in Document Analysis Systems, 2016.

F. Rayar, **T. Mondal**, S. Barrat, F. Bouali and G. Venturini **"Visual Analysis System for Word Features and Distances Qualitative Assessment"**, Submitted in Document Analysis Systems, 2016.

Bibliography

[Adamek and O'Connor, 2004] Adamek, T. and O'Connor, N. (2004). A Multiscale Representation Method for Nonrigid Shapes With a Single Closed Contour. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(5):742–753.

[Adamek et al., 2006] Adamek, T., O'Connor, N. E., and Smeaton, A. F. (2006). Word matching using single closed contours for indexing handwritten historical documents. *IJDAR*, 9(2-4):153–165.

[Agazzi and Kuo, 1994] Agazzi, O. and Kuo, S.-S. K. S.-S. (1994). Keyword spotting in poorly printed documents using pseudo 2-D hidden Markov models. *TPAMI*, 16(8):842–848.

[Agrawal et al., 1993] Agrawal, R., Faloutsos, C., and Swami, A. N. (1993). Efficient Similarity Search In Sequence Databases. pages 69–84.

[Al-Naymat et al., 2009] Al-Naymat, G., Chawla, S., and Taheri, J. (2009). SparseDTW: A novel approach to speed up dynamic time warping. *Conferences in Research and Practice in Information Technology Series*, 101(2007):117–127.

[Albrecht, 2009] Albrecht, T. (2009). Dynamic Time Warping (DTW). pages 69–85.

[Aldavert et al., 2013] Aldavert, D., Rusinol, M., Toledo, R., and Llados, J. (2013). Integrating Visual and Textual Cues for Query-by-String Word Spotting. In *12th International Conference on Document Analysis and Recognition*, pages 511–515. IEEE.

[Almazán et al., 2012] Almazán, J., Gordo, A., Fornés, A., and Valveny, E. (2012). Efficient Exemplar Word Spotting. *Procedings of the British Machine Vision Conference 2012*, pages 67.1—-67.11.

[Almazan et al., 2013] Almazan, J., Gordo, A., Fornes, A., and Valveny, E. (2013). Handwritten Word Spotting with Corrected Attributes. *ICCV*, pages 1017–1024.

[Antonio Cardone, Ra K. Gupta, 2003] Antonio Cardone, Ra K. Gupta, M. K. (2003). A survey of shape similarity assessment algorithms for product design and manufacturing applications. *Journal of Computing and Information Science in Engineering*, 3:109–118.

[Anurag Bhardwaj, Damien Jose, 2008] Anurag Bhardwaj, Damien Jose, V. G. (2008). Script Independent Word Spotting in Multilingual Documents. *Second International Workshop Cross Lingual Information Access*, pages 48–54.

[Attalla and Siy, 2005] Attalla, E. and Siy, P. (2005). Robust shape similarity retrieval based on contour segmentation polygonal multiresolution and elastic matching. *Pattern Recognition*, 38(12):2229–2241.

[Bai et al., 2009] Bai, S., Li, L., and Tan, C. L. (2009). Keyword Spotting in Document Images through Word Shape Coding. In *2009 10th International Conference on Document Analysis and Recognition*, pages 331–335. IEEE.

[Benedikt et al., 2008] Benedikt, L., Kajic, V., Cosker, D., Rosin, P. L., and Marshall, D. (2008). Facial Dynamics in Biometric Identification. In *Proceedings of the British Machine Vision Conference*.

[Cao et al., 2009] Cao, H., Bhardwaj, A., and Govindaraju, V. (2009). A probabilistic method for keyword retrieval in handwritten document images. *Pattern Recognition*, 42(12):3374–3382.

[Cao and Govindaraju, 2007] Cao, H. and Govindaraju, V. (2007). Template-free word spotting in low-quality manuscripts. *ICPR*, pages 1–5.

[Chan et al., 2006] Chan, J., Ziftci, C., and Forsyth, D. (2006). Searching off-line arabic documents. In *CVPR*, volume 2, pages 1455–1462.

[Charikar, 2002] Charikar, M. S. (2002). Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing - STOC '02*, page 380, New York, New York, USA. ACM Press.

[Chaudhuri and Pal, 1997] Chaudhuri, B. and Pal, U. (1997). An OCR system to read two Indian language scripts: Bangla and Devnagari (Hindi). *Proceedings of the Fourth International Conference on Document Analysis and Recognition*, 2:1011 – 1015.

[Chaudhuri and Pal, 1998] Chaudhuri, B. and Pal, U. (1998). A complete printed Bangla OCR system. *Pattern Recognition*, 31(5):531–549.

[Chen et al., 1995] Chen, F. R., Bloomberg, D. S., and Wilcox, L. D. (1995). Spotting phrases in lines of imaged text. In Vincent, L. M. and Baird, H. S., editors, *IS&T/SPIE's Symposium on Electronic Imaging: Science & Technology*, pages 256–269. International Society for Optics and Photonics.

[Chen, 2011] Chen, H. (2011). Robust Text Detection In Natural Images With Edge-Enhanced Maximally Stable Extremal Regions. In *Image Processing (ICIP), 18th IEEE International Conference on IEEE*.

[Chen et al., 2105] Chen, Y., Keogh, E., Hu, B., Begum, N., Bagnall, A., Mueen, A., and Batista, G. (2105). The UCR Time Series Classification Archive.

[Choudhary et al., 2013] Choudhary, A., Rishi, R., and Ahlawat, S. (2013). A New Character Segmentation Approach for Off-Line Cursive Handwritten Words. *Procedia Computer Science*, 17:88–95.

[Chu et al., 2002] Chu, S., Keogh, E., and Hart, D. (2002). Iterative Deepening Dynamic Time Warping for Time Series. *Time*, pages 195–212.

[Cristea, 2002] Cristea, P. D. (2002). Conversion of nucleotides sequences into genomic signals. *Journal of cellular and molecular medicine*, 6(2):279–303.

[Dalal and Triggs, 2005] Dalal, N. and Triggs, B. (2005). Histograms of Oriented Gradients for Human Detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893. IEEE.

[Das et al., 1997] Das, G., Gunopulos, D., and Mannila, H. (1997). *Principles of Data Mining and Knowledge Discovery*, volume 1263 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Dawson and Efford, 2009] Dawson, D. K. and Efford, M. G. (2009). Bird population density estimated from acoustic signals. *Journal of Applied Ecology*, 46(6):1201–1209.

[Dietrich et al., 2004] Dietrich, C., Palm, G., Riede, K., and Schwenker, F. (2004). Classification of bioacoustic time series based on the combination of global and local decisions. *Pattern Recognition*, 37(12):2293–2305.

[Ding et al., 2008] Ding, H., Trajcevski, G., Scheuermann, P., Wang, X., and Keogh, E. (2008). Querying and mining of time series data: experimental comparison of representations and distance measures. *Proceedings of the VLDB Endowment*, 1:1542–1552.

[Dovgalecs et al., 2013] Dovgalecs, V., Burnett, A., Tranouez, P., Nicolas, S., and Heutte, L. (2013). Spot It! Finding Words and Patterns in Historical Documents. In *ICDAR*, pages 1039–1043. IEEE.

[Eads et al., 2002] Eads, D., Hill, D., Davis, S., Perkins, S., Ma, J., Porter, R., and Theiler, J. (2002). Genetic Algorithms and Support Vector Machines for Time Series Classification. *Proceedings of SPIE*, 4787:74–85.

[Edwards, J. Teh, Y. W. Forsyth, D. Bock, R. Maire, M. Vesom, 2004] Edwards, J. Teh, Y. W. Forsyth, D. Bock, R. Maire, M. Vesom, G. (2004). Making latin manuscripts searchable using gHMM's. *Advances in Neural Information Processing Systems*, 17:385–392.

[Faloutsos et al., 1994] Faloutsos, C., Ranganathan, M., and Manolopoulos, Y. (1994). Fast subsequence matching in time-series databases. *ACM SIGMOD Record*, 23(2):419–429.

[Fernández et al., 2007] Fernández, S., Graves, A., and Schmidhuber, J. (2007). An application of recurrent neural networks to discriminative keyword spotting. *The 17th international conference on Artificial neural networks*, pages 220–229.

[Fernandez-Mota et al., 2014] Fernandez-Mota, D., Riba, P., Fornes, A., and Llados, J. (2014). On the Influence of Key Point Encoding for Handwritten Word Spotting. In *14th International Conference on Frontiers in Handwriting Recognition*, pages 476–481.

[Fischer et al., 2013] Fischer, A., Frinken, V., Bunke, H., and Suen, C. Y. (2013). Improving HMM-Based Keyword Spotting with Character Language Models. *ICDAR*, pages 506–510.

[Fischer et al., 2010a] Fischer, A., Keller, A., Frinken, V., and Bunke, H. (2010a). HMM-based Word Spotting in Handwritten Documents Using Subword Models. *ICPR*, pages 3416–3419.

[Fischer et al., 2012] Fischer, A., Keller, A., Frinken, V., and Bunke, H. (2012). Lexicon-free handwritten word spotting using character HMMs. *Pattern Recognition Letters*, 33(7):934–942.

[Fischer et al., 2010b] Fischer, A., Riesen, K., and Bunke, H. (2010b). Graph Similarity Features for HMM-Based Handwriting Recognition in Historical Documents. In *ICFHR*, pages 253–258. IEEE.

[Frinken et al., 2012] Frinken, V., Fischer, A., Manmatha, R., and Bunke, H. (2012). A novel word spotting method based on recurrent neural networks. *IEEE TPAMI*, 34(2):211–224.

[Garofolo et al., 1993] Garofolo, J., Lamel, L., Fisher, W., Fiscus, J., Pallett, D., Dahlgren, N., and Zue, V. (1993). TIMIT Acoustic-Phonetic Continuous Speech Corpus (TIMIT).

[Gatos et al., 2014] Gatos, B., Louloudis, G., Causer, T., Grint, K., Romero, V., Sánchez, J. A., Toselli, A. H., and Vidal, E. (2014). Ground-Truth production in the tranScriptorium project. In *11th IAPR International Workshop on Document Analysis Systems (DAS)*.

[Gatos and Pratikakis, 2009] Gatos, B. and Pratikakis, I. (2009). Segmentation-free Word Spotting in Historical Printed Documents. *ICDAR*, pages 271–275.

[Gatos et al., 2006] Gatos, B., Pratikakis, I., and Perantonis, S. (2006). Adaptive degraded document image binarization. *PR*, 39(3):317–327.

[Giotis et al., 2015] Giotis, A. P., Sfikas, G., Nikou, C., and Gatos, B. (2015). Shape-based Word Spotting in Handwritten Document Images. pages 561–565.

[Górecki, 2014] Górecki, T. (2014). Using derivatives in a longest common subsequence dissimilarity measure for time series classification. *Pattern Recognition Letters*, 45:99–105.

[Górecki and Luczak, 2014] Górecki, T. and Luczak, M. (2014). Non-isometric transforms in time series classification using DTW. *Knowledge-Based Systems*, 61:98–108.

[Górecki and Luczak, 2015] Górecki, T. and Luczak, M. (2015). Multivariate time series classification with parametric derivative dynamic time warping. *Expert Systems with Applications*, 42(5):2305–2312.

[Güler and Ubeyli, 2005] Güler, I. and Ubeyli, E. D. (2005). Adaptive neuro-fuzzy inference system for classification of EEG signals using wavelet coefficients. *Journal of neuroscience methods*, 148:113–121.

[Gullo et al., 2009] Gullo, F., Ponti, G., Tagarelli, A., and Greco, S. (2009). A time series representation model for accurate and fast similarity detection. *Pattern Recognition*, 42:2998–3014.

[Günter and Bunke, 2005] Günter, S. and Bunke, H. (2005). Off-line cursive handwriting recognition using multiple classifier systems on the influence of vocabulary, ensemble, and training set size. *Optics and Lasers in Engineering*, 43(3-5):437–454.

[Haralick and Shapiro, 1992] Haralick, R. M. and Shapiro, L. G. (1992). Computer and Robot Vision.

[Howe, 2013] Howe, N. R. (2013). Part-Structured Inkball Models for One-Shot Handwritten Word Spotting. *ICDAR*, pages 582–586.

[Hüsken and Stagge, 2003] Hüsken, M. and Stagge, P. (2003). Recurrent neural networks for time series classification. *Neurocomputing*, 50:223–235.

[Indyk and Motwani, 1998] Indyk, P. and Motwani, R. (1998). Approximate nearest neighbors. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing - STOC '98*, pages 604–613, New York, USA. ACM Press.

[Itakura, 1975] Itakura, F. (1975). Minimum prediction residual principle applied to speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 23(1):67–72.

[Jagadish and Faloutsos, 1998] Jagadish, H. and Faloutsos, C. (1998). Efficient retrieval of similar time sequences under time warping. In *ICDE*, pages 201–208. IEEE Comput. Soc.

[Jalba et al., 2005] Jalba, A. C., Wilkinson, M. H. F., Roerdink, J. B. T. M., Bayer, M. M., and Juggins, S. (2005). Automatic diatom identification using contour analysis by morphological curvature scale spaces. *Machine Vision and Applications*, 16(4):217–228.

[Jawahar et al., 2004] Jawahar, C. V., Meshesha, M., and Balasubramanian, A. (2004). Searching in Document Images. *ICVGIP*, pages 622–627.

[Jeong et al., 2011] Jeong, Y. S., Jeong, M. K., and Omitaomu, O. A. (2011). Weighted dynamic time warping for time series classification. In *Pattern Recognition*, volume 44, pages 2231–2240. Elsevier.

[Jiahong Yuan, 2008] Jiahong Yuan, M. L. (2008). Speaker identification on the SCOTUS corpus. *Proceedings of Acoustics*.

[Keogh et al., 2001] Keogh, E., Chakrabarti, K., Pazzani, M., and Mehrotra, S. (2001). Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases. *Knowledge and Information Systems*, 3(3):263–286.

[Keogh and Kasetty, 2003] Keogh, E. and Kasetty, S. (2003). On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration. *Data Mining and Knowledge Discovery*, 7(4):349–371.

[Keogh and Ratanamahatana, 2005] Keogh, E. and Ratanamahatana, C. A. (2005). Exact indexing of dynamic time warping. *Knowledge and Information Systems*, 7(3):358–386.

[Keogh and Pazzani, 2000] Keogh, E. J. and Pazzani, M. J. (2000). Scaling up dynamic time warping for datamining applications. *KDD*, pages 285–289.

[Keshet et al., 2009] Keshet, J., Grangier, D., and Bengio, S. (2009). Discriminative keyword spotting. *Speech Communication*, 51(4):317–329.

[Kessentini et al., 2013] Kessentini, Y., Chatelain, C., and Paquet, T. (2013). Word Spotting and Regular Expression Detection in Handwritten Documents. In *ICDAR*, pages 516–520. IEEE.

[Kessentini and Paquet, 2015] Kessentini, Y. and Paquet, T. (2015). Keyword spotting in handwritten documents based on a generic text line HMM and a SVM verification. In *ICDAR*.

[Khurshid et al., 2012] Khurshid, K., Faure, C., and Vincent, N. (2012). Word spotting in historical printed documents using shape and sequence comparisons. *Pattern Recognition*, 45(7):2598–2609.

[Kolcz et al., 2000] Kolcz, A., Alspector, J., and Augusteijn, M. (2000). A line-oriented approach to word spotting in handwritten documents. *Pattern Analysis & Applications*, 3(2):153–168.

[Kulbacki et al., 2002] Kulbacki, M., Kulbacki, M., Segen, J., Segen, J., Bak, A., and Bak, A. (2002). Unsupervised Learning Motion Models Using Dynamic Time Warping. *Systems Research*, (July 2015):1–10.

[Kulis and Grauman, 2009] Kulis, B. and Grauman, K. (2009). Kernelized locality-sensitive hashing for scalable image search. *2009 IEEE 12th International Conference on Computer Vision*, (Iccv):2130–2137.

[Latecki et al., 2007a] Latecki, L. J., Koknar-tezel, S., Wang, Q., and Megalooikonomou, V. (2007a). Sequence Matching Capable of Excluding Outliers. In *Int. Conf. on Knowledge Discovery and Data Mining (KDD)*.

[Latecki et al., 2007b] Latecki, L. J., Megalooikonomou, V., Wang, Q., and Yu, D. (2007b). An elastic partial shape matching technique. *Pattern Recognition*, 40(11):3069–3080.

[Latecki et al., 2007c] Latecki, L. J., Wang, Q., Koknar-Tezel, S., and Megalooikonomou, V. (2007c). Optimal Subsequence Bijection. *ICDM*, pages 565–570.

[Lavrenko et al., 2004] Lavrenko, V., Rath, T., and Manmatha, R. (2004). Holistic word recognition for handwritten historical documents. *First International Workshop on Document Image Analysis for Libraries, Proceedings.*, pages 278–287.

[Lew et al., 2006a] Lew, M., Sebe, N., Djeraba, C., and Jain, R. (2006a). Content-based multimedia information retrieval: State of the art and challenges. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, 2(1):1–19.

[Lew et al., 2006b] Lew, M. S., Sebe, N., Djeraba, C., and Jain, R. (2006b). Content-based multimedia information retrieval. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 2(1):1–19.

[Leydier et al., 2005] Leydier, Y., Le Bourgeois, F., and Emptoz, H. (2005). Omnilingual segmentation-free word spotting for ancient manuscripts indexation. In *ICDAR*, volume 2005, pages 533–537 Vol. 1. IEEE.

[Leydier et al., 2007] Leydier, Y., Lebourgeois, F., and Emptoz, H. (2007). Text search for medieval manuscript images. *Pattern Recognition*, 40(12):3552–3567.

[Li and Yang, 2013] Li, H. and Yang, L. (2013). Accurate and Fast Dynamic Time Warping. *Advanced Data Mining and Applications*, pages 133–144.

[Lin-Lin, 2009] Lin-Lin, L. (2009). *Extraction of Textual Information from Images for Information Retrieval*. PhD thesis, National University of Singapore.

[Listgarten et al., 2005] Listgarten, J., Neal, R. M., Roweis, S. T., and Emili, A. (2005). Multiple Alignment of Continuous Time Series. *Advances in Neural Information Processing Systems*, 17(17):817–824.

[Lu et al., 2008] Lu, S., Li, L., and Tan, C. L. (2008). Document image retrieval through word shape coding. *TPAMI*, 30(11):1913–8.

[Lu and Tan, 2008] Lu, S. and Tan, C. L. (2008). Retrieval of machine-printed Latin documents through Word Shape Coding. *Pattern Recognition*, 41(5):1799–1809.

[Lu and Tan, 2002] Lu, Y. and Tan, C. L. (2002). Word spotting in Chinese document images without layout analysis. In *International Conference on Pattern Recognition*, volume 3, pages 57–60. IEEE Comput. Soc.

[Lu and Tan, 2004] Lu, Y. and Tan, C. L. (2004). Information retrieval in document image databases. *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1398–1410.

[Manmatha, R. Chengfeng and Riseman, 1996] Manmatha, R. Chengfeng, H. and Riseman, E. (1996). Word spotting: a new approach to indexing handwriting. In *CVPR*, pages 631–637. IEEE Comput. Soc. Press.

[Manthalkar et al., 2003] Manthalkar, R., Biswas, P., and Chatterji, B. (2003). Rotation and scale invariant texture features using discrete wavelet packet transform. *Pattern Recognition Letters*, 24(14):2455–2462.

[Marti and Bunke, 2001] Marti, U. and Bunke, H. (2001). Using a statistical language model to improve the performance of an HMM-based cursive handwriting recognition system. *International Journal of Pattern Recognition and ...*, 15:65–90.

[Mayer and Zinke, 2006] Mayer, D. and Zinke, A. (2006). Iterative multi scale dynamic time warping. *Universität Bonn, CG-2006/1*.

[Meshesha and Jawahar, 2008a] Meshesha, M. and Jawahar, C. V. (2008a). Matching word images for content-based retrieval from printed document images. *International Journal of Document Analysis ...*, 11(1):29–38.

[Meshesha and Jawahar, 2008b] Meshesha, M. and Jawahar, C. V. (2008b). Matching word images for content-based retrieval from printed document images. *IJDAR*, 11(1):29–38.

[Moghaddam and Cheriet, 2009] Moghaddam, R. F. and Cheriet, M. (2009). Application of Multi-Level Classifiers andClustering for Automatic Word Spotting in Historical Document Images. *ICDAR*, pages 511–515.

[Mondal et al.,] Mondal, T., Ragot, N., and Ramel, J.-y. Flexible Sequence Matching Technique : Application to Word Spotting in Degraded Documents. *2014 14th International Conference on Frontiers in Handwriting Recognition*.

[Mondal et al., 2015a] Mondal, T., Ragot, N., and Ramel, J.-y. (2015a). Exemplary Sequence Cardinality : An Effective Application for Word Spotting. In *ICDAR*.

[Mondal et al., 2014] Mondal, T., Ragot, N., Ramel, J.-y., and Pal, U. (2014). Flexible Sequence Matching Technique: Application to Word Spotting in Degraded Documents. In *ICFHR*, pages 210–215. IEEE.

[Mondal et al., 2015b] Mondal, T., Ragot, N., Ramel, J.-Y., and Pal, U. (2015b). Performance Evaluation of DTW and its Variants for Word Spotting in Degraded Documents. In *ICDAR*.

[Mondal et al., 2009] Mondal, T., Ragot, N., Ramel, J.-y., Pal, U., Rabelais, U. F., and Informatique, L. (2009). A Fast Word Retrieval Technique Based on Kernelized Locality Sensitive Hashing ïĆğ A fast word retrieval approach based on KLSH , which defines hash functions using arbitrary kernel , which are locality sensitive , thereby permitting sub linear time approxi. 139(2007):4700.

[Munich and Perona, 1999] Munich, M. and Perona, P. (1999). Continuous dynamic time warping for translation-invariant curve alignment with applications to signature verification. *ICCV*, 1:108–115.

[Nagendar and Jawahar, 2015] Nagendar, G. and Jawahar, C. V. (2015). Efficient Word Image Retrieval using Fast DTW Distance.

[Nakayama, 1994] Nakayama, T. (1994). Modeling Content Identification from Document Images. *Proceedings Fourth Conference Applied Natural Language Processing (ANLP ',94)*, pages 22–27.

[Nakayama, 1996] Nakayama, T. (1996). Content-Oriented Categorization of Document Images. In *Proceedings of the 16th International Conference on Computational Linguistics*, pages 818–823.

[Niennattrakul and Ratanamahatana, 2007] Niennattrakul, V. and Ratanamahatana, C. A. (2007). On Clustering Multimedia Time Series Data Using K-Means and Dynamic Time Warping. In *2007 International Conference on Multimedia and Ubiquitous Engineering (MUE'07)*, pages 733–738. IEEE.

[Nikolaou et al., 2010] Nikolaou, N., Makridis, M., Gatos, B., Stamatopoulos, N., and Papamarkos, N. (2010). Segmentation of historical machine-printed documents using Adaptive Run Length Smoothing and skeleton segmentation paths. *Image and Vision Computing*, 28(4):590–604.

[Nobuyuki, 1979] Nobuyuki, O. (1979). A Threshold Selection Method from Gray-Level Histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66.

[Oka, 1998] Oka, R. (1998). Spotting method for classification of real world data. *The Computer Journal*, 41(8):1–6.

[Pal et al., 2012] Pal, U., Jayadevan, R., and Sharma, N. (2012). Handwriting Recognition in Indian Regional Scripts. *ACM Transactions on Asian Language Information Processing*, 11(1):1–35.

[Perronnin and Rodriguez-Serrano, 2009] Perronnin, F. and Rodriguez-Serrano, J. a. (2009). Fisher Kernels for Handwritten Word-spotting. *ICDAR*, pages 106–110.

[Plamondon and Srihari, 2000] Plamondon, R. and Srihari, S. (2000). Online and off-line handwriting recognition: a comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):63–84.

[Prasad et al., 2001] Prasad, B. G., Gupta, S. K., and Biswas, K. K. (2001). Color and Shape Index for Region-Based Image Retrieval. pages 716–728.

[R. Manmatha, 2003] R. Manmatha, T. M. R. (2003). Indexing of Handwritten Historical Documents - Recent Progress. *Proceedings of Symposium on Document Image Understanding Technology (SDIUT)*, pages 77–85.

[Rama, 2013] Rama, T. (2013). Phonotactic diversity predicts the time depth of the world's language families. *PloS one*, 8(5):e63238.

[Ratanamahatana, 2005] Ratanamahatana, C. (2005). *Improving efficiency and effectiveness of dynamic time warping in large time series databases*. PhD thesis, UNIVERSITY OF CALIFORNIA RIVERSIDE.

[Ratanamahatana and Keogh, 2004a] Ratanamahatana, C. and Keogh, E. (2004a). Everything you know about dynamic time warping is wrong. *Third Workshop on Mining Temporal and Sequential Data*, pages 22–25.

[Ratanamahatana and Keogh, 2005] Ratanamahatana, C. A. and Keogh, E. (2005). Three myths about dynamic time warping data mining. In *Proceedings of SIAM International Conference on Data Mining (SDM'05)*, pages 506–510.

[Ratanamahatana and Keogh, 2004b] Ratanamahatana, C. A. C. and Keogh, E. (2004b). Making time-series classification more accurate using learned constraints. *Proceedings of SIAM*, pages 11–22.

[Rath and Manmatha, 2003] Rath, T. M. and Manmatha, R. (2003). Word image matching using dynamic time warping. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2:521–527.

[Rath and Manmatha, 2006] Rath, T. M. and Manmatha, R. (2006). Word spotting for historical documents. *IJDAR*, 9(2-4):139–152.

[Rath et al., 2004] Rath, T. M., Manmatha, R., and Lavrenko, V. (2004). A Search Engine for Historical Manuscript Images. In *ACM SIGIR*, pages 369–376.

[Riba and Llad, 2015] Riba, P. and Llad, J. (2015). Handwritten Word Spotting by Inexact Matching of Grapheme Graphs. pages 781–785.

[Rodriguez and Perronnin, 2008] Rodriguez, J. and Perronnin, F. (2008). Unsupervised writer style adaptation for handwritten word spotting. *ICPR*, pages 1–4.

[Rodríguez-Serrano and Perronnin, 2009] Rodríguez-Serrano, J. a. and Perronnin, F. (2009). Handwritten word-spotting using hidden Markov models and universal vocabularies. *Pattern Recognition*, 42(9):2106–2116.

[Rothacker et al., 2013] Rothacker, L., Rusinol, M., and Fink, G. a. (2013). Bag-of-Features HMMs for Segmentation-Free Word Spotting in Handwritten Documents. *ICDAR*, pages 1305–1309.

[Rothfeder et al., 2003] Rothfeder, J. L., Feng, S., and Rath, T. M. (2003). Using Corner Feature Correspondences to Rank Word Images by Similarity. *2003 Conference on Computer Vision and Pattern Recognition Workshop*, 3.

[Roverso, 2000] Roverso, D. (2000). Multivariate Temporal Classification By Windowed Wavelet Decomposition And Recurrent Neural Networks. In *3rd ANS International Topical Meeting on Nuclear Plant Instrumentation, Control and Human-Machine Interface.*

[Roy et al., 2013] Roy, U., Sankaran, N., Sankar, K. P., and Jawahar, C. (2013). Character N-Gram Spotting on Handwritten Documents Using Weakly-Supervised Segmentation. In *ICDAR*, pages 577–581. IEEE.

[Rusiñol et al., 2011] Rusiñol, M., Aldavert, D., Toledo, R., and Lladós, J. (2011). Browsing Heterogeneous Document Collections by a Segmentation-Free Word Spotting Method. *2011 International Conference on Document Analysis and Recognition*, pages 63–67.

[Rusiñol et al., 2015] Rusiñol, M., Aldavert, D., Toledo, R., and Lladós, J. (2015). Efficient segmentation-free keyword spotting in historical document collections. *Pattern Recognition*, 48(2):545–555.

[Sakoe and Chiba, 1978] Sakoe, H. and Chiba, S. (1978). Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49.

[Salvador and Chan, 2007] Salvador, S. and Chan, P. (2007). FastDTW : Toward Accurate Dynamic Time Warping in Linear Time and Space. *Intell. Data Anal.*, 11(5):561–580.

[Sarkar, 2013] Sarkar, S. (2013). *Word Spotting in Cursive Handwritten Documents Using Modified Character Shape Codes*, volume 178 of *Advances in Intelligent Systems and Computing*. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Saykol et al., 2004] Saykol, E., Sinop, A. K., Güdükbay, U., Ulusoy, O., and Cetin, a. E. (2004). Content-based retrieval of historical Ottoman documents stored as textual images. *TIP*, 13(3):314–25.

[Senior and Robinson, 1998] Senior, A. and Robinson, A. (1998). An off-line cursive handwriting recognition system. *TPAMI*, 20(3):309–321.

[Shijian and Lim Tan, 2008] Shijian, L. and Lim Tan, C. (2008). Script and language identification in noisy and degraded document images. *IEEE transactions on pattern analysis and machine intelligence*, 30(1):14–24.

[Shijian Lu, Chew Lim Tan, 2006] Shijian Lu, Chew Lim Tan, W. H. (2006). Script and language identification in degraded and distorted document images. *National Conference on Artificial Intelligence*, pages 769–774.

[Smeaton and Spitz, 1997] Smeaton, A. and Spitz, A. (1997). Using character shape coding for information retrieval. In *Proceedings of the Fourth International Conference on Document Analysis and Recognition*, volume 2, pages 974–978. IEEE Comput. Soc.

[Smolinski et al., 2008] Smolinski, T. G., Milanova, M. G., and Hassanien, A.-E., editors (2008). *Applications of Computational Intelligence in Biology*, volume 122 of *Studies in Computational Intelligence*. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Spitz, 1997] Spitz, A. (1997). Determination of the script and language content of document images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(3):235–245.

[Srihari et al., 2006] Srihari, S., Srinivasan, H., Babu, P., and Bhole, C. (2006). Spotting words in handwritten Arabic documents. In *Electronic Imaging*, pages 606702–606702–12. International Society for Optics and Photonics.

[Stamatopoulos et al., 2010] Stamatopoulos, N., Gatos, B., and Georgiou, T. (2010). Page frame detection for double page document images. *DAS*, pages 401–408.

[Sudholt et al., 2015] Sudholt, S., Rothacker, L., and Fink, G. A. (2015). Learning Local Image Descriptors for Word Spotting. pages 651–655.

[Tak, 2008] Tak, Y.-S. (2008). Pruning and Matching Scheme for Rotation Invariant Leaf Image Retrieval. *KSII Transactions on Internet and Information Systems*, 2(6):280–298.

[Tan et al., 2002] Tan, C. L., Huang, W., Yu, Z., and Xu, Y. (2002). Imaged document text retrieval without OCR. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(6):838–844.

[Tan et al., 2003] Tan, C. L. I. M., Huang, W., Sung, S. A. M. Y., Yu, Z., and Xu, Y. I. (2003). Text Retrieval from Document Images Based on Word Shape Analysis. *Applied Intelligence*, 18(3):257–270.

[Tarafdar et al., 2010] Tarafdar, A., Mondal, R., Pal, S., Pal, U., and Kimura, F. (2010). Shape Code Based Word-Image Matching for Retrieval of Indian Multi-lingual Documents. In *ICPR*, pages 1989–1992. IEEE.

[Terasawa and Tanaka, 2009] Terasawa, K. and Tanaka, Y. (2009). Slit Style HOG Feature for Document Image Word-Spotting. *ICDAR*, pages 116–120.

[Torralba et al., 2008] Torralba, A., Fergus, R., and Weiss, Y. (2008). Small codes and large image databases for recognition. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE.

[Toselli and Vidal, 2013] Toselli, A. H. and Vidal, E. (2013). Fast HMM-Filler Approach for Key Word Spotting in Handwritten Documents. *ICDAR*, pages 501–505.

[Trifa et al., 2007] Trifa, V., Girod, L., Collier, T. C., Blumstein, D., and Taylor, C. E. (2007). Automated Wildlife Monitoring Using Self-Configuring Sensor Networks Deployed in Natural Habitats. *Center for Embedded Network Sensing*.

[Übeyli, 2008] Übeyli, E. D. (2008). Wavelet/mixture of experts network structure for EEG signals classification. *Expert Systems with Applications*, 34(3):1954–1962.

[Vasilopoulos and Kavallieratou, 2013] Vasilopoulos, N. and Kavallieratou, E. (2013). A classification-free word-spotting system. In *SPIE*, volume 8658, page 86580F.

[Vincenzo, 2002] Vincenzo, P. (2002). Variation in the function of Eagle Owl vocal behaviour: territorial defence and intra-pair communication? *Ethology, Ecology & Evolution*, 14(3):275–281.

[Vlachos et al., 2003] Vlachos, M., Hadjieleftheriou, M., Gunopulos, D., and Keogh, E. J. (2003). Indexing multi-dimensional time-series with support for multiple distance measures. In *KDD*, pages 216–225, New York, USA. ACM Press.

[Vlachos et al., 2002] Vlachos, M., Kollios, G., and Gunopulos, D. (2002). Discovering similar multidimensional trajectories. In *ICDE*, pages 673–684. IEEE Comput. Soc.

[Wakuya and Shida, 2002] Wakuya, H. and Shida, K. (2002). Time Series Prediction with Neural Network Model Based on the Bi-directional Computation Style: An Analytical Study and Its Estimation on Acquired Signal Transformation. *Transacition IEE; Japan*, 122-C(10):1794–1802.

[Wang et al., 2014a] Wang, P., Eglin, V., Garcia, C., Largeron, C., Llados, J., and Fornes, A. (2014a). A Novel Learning-Free Word Spotting Approach Based on Graph Representation. In *11th IAPR International Workshop on Document Analysis Systems*, pages 207–211. IEEE.

[Wang et al., 2014b] Wang, P., Veronique, E., Christine, L., Josep, L., and Alicia, F. (2014b). A Novel Learning-free Word Spotting Approach Based On Graph Representation. In *DAS*, pages 207–211.

[Wang et al., 2014c] Wang, Q.-F., Yin, F., and Liu, C.-L. (2014c). Unsupervised language model adaptation for handwritten Chinese text recognition. *Pattern Recognition*, 47:1202–1216.

[Wang et al., 2002] Wang, X., Ding, X., and Liu, C. (2002). Optimized Gabor filter based feature extraction for character recognition. In *Object recognition supported by user interaction for service robots*, volume 4, pages 223–226. IEEE Comput. Soc.

[Wei and Keogh, 2006] Wei, L. and Keogh, E. (2006). Semi-supervised time series classification. *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '06*, pages 748–753.

[Weiss et al., 2009] Weiss, Y., Torralba, A., and Fergus, R. (2009). Spectral Hashing. In *Advances in Neural Information Processing Systems*, pages 1753–1760.

[Wollmer et al., 2009] Wollmer, M., Eyben, F., Keshet, J., Graves, A., and Rigoll, G. (2009). Robust Discriminative Keyword Spotting for Emotionally Colored Spontaneous Speech Using Bidirectional LSTM Networks. *ICASSP*, pages 3949 – 3952.

[Xi et al., 2006] Xi, X., Keogh, E., Shelton, C., Wei, L., and Ratanamahatana, C. A. (2006). Fast time series classification using numerosity reduction. In *Proceedings of the 23rd international conference on Machine learning (ICML)*, pages 1033—-1040.

[Yang and Lee,] Yang, T. and Lee, D. T3 : On Mapping Text To Time Series. In *Proceedings of the 3rd Alberto Mendelzon International Workshop on Foundations of Data Management*, Arequipa, Peru.

[Yao et al., 2015] Yao, S., Wen, Y., and Lu, Y. (2015). HoG based Two-Directional Dynamic Time Warping for Handwritten Word Spotting. In *ICDAR*.

[Yau et al., 2003] Yau, S. S.-T., Wang, J., Niknejad, A., Lu, C., Jin, N., and Ho, Y.-K. (2003). DNA sequence representation without degeneracy. *Nucleic acids research*, 31(12):3078–80.

[Yu et al., 2007] Yu, F. Y. F., Dong, K. D. K., Chen, F. C. F., Jiang, Y. J. Y., and Zeng, W. Z. W. (2007). Clustering Time Series with Granular Dynamic Time Warping Method. *2007 IEEE International Conference on Granular Computing (GRC 2007)*.

[Zhang et al., 2003] Zhang, B., Srihari, S. N., and Huang, C. (2003). Word image retrieval using binary features. In *Proceedings of the SPIE*, volume 5296, pages 45–53.

[Zhang and Tan, 2013] Zhang, X. and Tan, C. L. (2013). Segmentation-Free Keyword Spotting for Handwritten Documents Based on Heat Kernel Signature. In *ICDAR*, pages 827–831. IEEE.

[Zifan et al., 2007] Zifan, A., Saberi, S., Moradi, M. H., and Towhidkhah, F. (2007). Automated ECG Segmentation Using Piecewise Derivative Dynamic Time Warping. *International Journal of Medical, Health, Pharmaceutical and Biomedical Engineering*, 1(3):764–768.