Language-neutral algorithms for partial search on online handwritten text

A PROJECT REPORT

SUBMITTED IN PARTIAL FULFILMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

Master of Engineering

IN

COMPUTER SCIENCE AND ENGINEERING

by

R. Sriraghavendra

Computer Science and Automation

Indian Institute of Science

BANGALORE – 560 012

July 2007

Dedicated to

My parents & grandparents

Acknowledgements

I am immensely thankful to my advisor, Dr. Chiranjib Bhattacharyya, for his guidance, for being a constant source of motivation and for all the help and support that he has rendered to me. I would also like to thank him for all that he has taught me in machine learning and allied areas.

I am very thankful to Karthik.K for providing me invaluable help in in my project. I am also very thankful to Dr. Subrahmanya and Dr, Swami Manohar of Picopeta for their help and encouragement.

I would like to thank Profs. M.Narasimha Murty, Y.Narahari and S.K.Shevade for giving several suggestions for improvement.

I thank all members of the machine learning lab for providing an excellent, intellectually stimulating environment to work in. Many thanks to all my friends who kept motivating me!

I am very very thankful to my family - my parents, grandparents and sister, for their support and encouragement throughout the course of my studies.

Abstract

We propose novel language-neutral, dynamic programming based algorithms for performing partial search on online handwritten text documents. These algorithms enable querying online multilingual handwritten documents with substrings of words rather than just whole words. The problem of searching for a query string is posed as one of matching two curves in a two-dimensional Euclidean space. Assuming a writer-specific setting, we present two new approaches for this purpose, one based on the Fréchet distance for curves, and the other based on a pairwise Hidden Markov model. For the Fréchet distance based approach, we formulate variants of the discrete Fréchet distance for performing partial search. We also formulate similar variants of the well-known dynamic time warping (DTW) distance and compare the the performance of the Fréchet distance based variants against those based on DTW. For the second approach, we formulate a Pair HMM to model partial matching of curves, i.e. alignment of one curve as a subcurve of another. We evaluated both approaches by carrying out extensive experiments on a datagroup from the open source UNIPEN dataset[1], consisting of over 16,000 words written by 7 users. We also implemented a search application based on the the Fréchet-DTW approach for searching free-form, multilingual Paper documents in a Simputer. Results obtained using this search application indicate that our search algorithms can perform well in multilingual settings as well and hence suited for PDAs.

[1] http://unipen.nici.ru.nl/cdroms/

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

With the growing popularity of mobile devices worldwide, there is an increasing need for providing input interface in local languages. Designing a conventional input interface, such as a keypad (physical or onscreen), is a major challenge in case of languages that have complex character sets; this is the case for several Asian languages. *Digital ink technology*, i.e. storing handwritten input in its original raw format [6], allows handwritten data to contain text written in *any language*, drawings and other free-form symbols, not just text in languages for which handwriting recognition is supported. This technology can be used in when users do not require that their handwritten data be recognized by the device, for instance, when the device is used for *digital note-taking* and the user only wants the notes in the original form.

In this project, we consider the problem of searching online handwritten documents available as time series in digital ink format for occurrences of a *handwritten search term* (query string). The documents being searched are assumed to have been *written by the same user* who has written the search query. Alternately, we may want to retrieve documents that have an occurrence of the query string. There are some major challenges to be addressed in designing algorithms for this problem, especially in the context of using them for search on PDAs in a multilingual setting:

- Handling multilingual handwritten documents: It is extremely difficult to develop reliable handwriting recognition systems for all possible languages that can be present in

a multilingual document. Besides, it is not possible to support a large number of handwriting recognition systems in mobile devices. The search algorithm should therefore not depend on handwriting recognition - it should be language-neutral

- Limited computing resources: Mobile devices typically have limited computing resources when compared to standard computers. The algorithm should be capable of yielding an efficient implementation in this setting.

- Variability in handwriting: The algorithm should be robust to minor changes in shape of characters/words arising due to variability in handwriting of the user.

The main contributions of this project are two approaches for performing partial search on online handwritten data. As noted before, the problem is essentially equivalent to matching subcurves. We propose new variants of the Fréchet distance for curves and correspondingly analogous variants of the DTW. We derive dynamic programming (DP) based algorithms for computing these variants. It should be noted that to the best of our knowledge, there is no previous work that explored the use of Fréchet distance in the context of handwriting search or recognition. The other major contribution of this project is a new pairwise Hidden Markov model (PHMM) for partial matching of curves; the model is motivated by probabilistic models of sequence alignment [3] used in computational biology. A DP-based algorithm is derived using the Viterbi scheme for inferring the match based on this PHMM. Experimental results on the open source UNIPEN dataset [5] show that the PHMM-based algorithm outperforms the other proposed algorithms.

The rest of this report is organized as follows. In chapter 2 we present the Fréchet distance based approach that we propose for partial search; we first discusses the concept of Fréchet distance and existing work on the discrete Fréchet distance. We then present the new variants of Fréchet and DTW distances and algorithms to compute these. In chapter 3 we discuss our PHMM-based approach for partial search and then present our language-neutral search technique. In chapter 4, we provide the details of our experiments, discuss the results obtained and also discuss the applications we developed based on the proposed approaches. We conclude in chapter 5 by discussing some possible future work.

Chapter 2

Fréchet distance and variants

2.1 Curve matching for multilingual search

We pose the problem of searching for a handwritten query in a handwritten document as one of searching for a given geometric pattern in a collection of geometric shapes. The geometric shapes are simply the curves representing the shape of handwritten words and are defined by the sample points in those words. Searching involves finding the similarity between the given curve and the curves corresponding to words in the handwritten document. This is essentially a problem of *curve matching*. In this project, we explore the use of Fréchet distance and its variants for curve matching in handwriting search. For the purpose of benchmarking these distance measures, we also consider the the well-known Dynamic Time Warping (DTW) distance [10] and formulate variants of DTW that are analogous to the Fréchet distance variants.

2.2 Preliminaries

In this section we discuss some preliminary definitions and notations used in the rest of the chapter. We borrow many of these notations from [2] and [8].

For any two integers a and b, $[a : b]$ denotes $\{a, a + 1, ..., b\}$, the set of all integers between a and b.

$V = \mathbb{R}^k$ denotes a k-dimensional Euclidean vector space.

Norms of sequence of vectors: For $f \in V^I$ $(I < \infty)$

$$\left\|f\right\|_\infty = \sup_{t \in I} \left\|f(t)\right\|_2$$
$$\left\|f\right\|_2 = \sqrt{\sum_{t \in I} \left\|f(t)\right\|_2^2}$$

Curve: A *(parameterized) curve* in $\mathbb{R}^k$ is defined as a continuous mapping $f : [a, b] \mapsto \mathbb{R}^k$, where $a, b \in \mathbb{R}$

Polygonal curve: A *(parameterized) polygonal curve* of length $m \in \mathbb{N}$ in $\mathbb{R}^k$ is defined as a mapping $P : [0, m] \mapsto \mathbb{R}^k$ such that

$$P(i + \lambda) = (1 - \lambda)P(i) + \lambda P(i + 1) \;\; \forall \lambda \in [0, 1]$$

In other words, a polygonal curve of length m is a piecewise linear curve consisting of m line segments and is defined completely by the $m+1$ points $p_0 = P(0), p_1 = P(1), ..., p_m = P(m)$ that form the line segments $p_0p_1, p_1p_2, ..., p_{m-1}p_m$. Hence the polygonal curve P can also defined as a mapping based on *discrete* parameterization, $P : [0 : m] \mapsto \mathbb{R}^k$.

Monotonic path: A *monotonic path* of length K is defined as a mapping $\pi \in (\mathbb{Z} \times \mathbb{Z})^{[0:K]}$ with the property that $\pi(i) - \pi(i - 1) \in (1, 0), (0, 1), (1, 1)$ for all $i \in [1 : K]$.

Weighted Euclidean distance: The *weighted Euclidean distance*, with weights $w_1 \geq 0$ and $w_2 \geq 0$ respectively, between two points (a_1, b_1) and (a_2, b_2) is defined as

$$\sqrt{w_1 * (a_1 - a_2)^2 + w_2 * (b_1 - b_2)^2}$$

2.3 Fréchet distance

Fréchet distance [1, 2, 4, 8, 11] is measure of similarity between two curves. More precisely it is a dissimilarity measure since it is a distance - a smaller distance indicates a higher degree of similarity, with a distance value of 0 corresponding to exact match. It can be defined intuitively by considering the example of a man and a dog, each of whom is positioned at the start of one

of the two given curves. The man holds the (elastic) leash that the dog is tied to. At time 0, the man and the dog start walking on their respective curves towards the respective endpoints. Both the man and dog are constrained to move (monotonically) forward along the curve but they can move at arbitrary speeds. They are not allowed to jump over and skip any point of their respective curves. A motion of dog and man is considered to be *legal* if it satisfies these constraints. The Fréchet distance between the two curves is the the minimum over all trajectories (legal motions that take the man and the dog to their respective endpoints) of the maximum leash length needed for a fixed trajectory. Fréchet distance is sometimes also known as the "dog-man" distance. Since Fréchet distance preserves the notion of continuity along curves, it is well suited to measuring curve similarity. On the other hand, Hausdorff distance [11], another well-known distance measure used in curve matching, is defined over point sets (rather than on curves) and does not reflect curve similarity well in many cases [2, 11].

2.3.1 Formal definition

The *Fréchet distance* (also called the *continuous Fréchet distance* to distinguish it from its discrete variant) between two curves $P : [0, m] \mapsto \mathbb{R}^k$ and $Q : [0, n] \mapsto \mathbb{R}^k$, denoted by $\mathbf{F}(P, Q)$ is defined as

$$\mathbf{F}(P,Q) = \min_{\alpha:[0,1]\mapsto[0,m],\ \beta:[0,1]\mapsto[0,n]} \max_{t\in[0,1]} \left\| P(\alpha(t)) - Q(\beta(t)) \right\|_2 \tag{2.1}$$

where α and β range over all *continuous,* monotonically increasing functions of the form $\alpha : [0, 1] \mapsto [0, m]$ and $\beta : [0, 1] \mapsto [0, n]$ respectively.

With respect to the dog-man example, α and β define a trajectory and specify the position of the man and the dog respectively at each time instant t, which ranges from 0 to 1. Thus it is assumed that the man and the dog start at time instant 0 and both are at their respective endpoints at time instant 1. For a given trajectory (α, β), the (minimum) length of the leash required is equal to the maximum distance between the man and the dog attained on that trajectory, which is given by the inner max (over t) term in equation (2.1). The outer min term minimizes this length over all possible trajectories.

2.3.2 Computing the Fréchet distance

Alt and Godau [1] present an algorithm for computing the Fréchet distance between polygonal curves. For the *decision problem* of finding whether the Fréchet distance between two polygonal curves of length m and n is smaller than a given constant, they give an $O(mn)$ time algorithm. Based on this algorithm and the 'parametric search' technique, they derive an algorithm that computes the Fréchet distance between two curves in $O(mn\,log(mn))$ time. [9] gives an algorithm for computing the Fréchet distance between the more general class of *piecewise smooth curves*, curves that consist of a sequence of smooth curve pieces such as circular or parabolic arcs. If the pieces are algebraic of degree bounded by a constant, then this algorithm too has a time complexity of $O(mn\,log(mn))$.

2.4 Discrete Fréchet distance

Although the above mentioned algorithms for computing Fréchet distance have a moderately low asymptotic complexity ($O(mnlog(mn))$), they are not really practical since the parametric search technique used in these algorithms makes use of a sorting network with very high constants in the running time [11]. Using a simpler sorting algorithm leads to a higher asymptotic running time ($O(mn(log\ mn)^3)$ in case of polygonal curves) and still the parametric search is not easy to implement [11]. This is a major drawback of using the continuous Fréchet distance and motivates us to look for approximations or variants that can be computed faster using simpler algorithms. The *discrete Fréchet distance* [8, 2] is one such variant of the Fréchet distance and is defined for polygonal curves.

2.4.1 Formal definition

The *discrete Fréchet distance* between two *polygonal* curves $P : [0 : m] \mapsto \mathbb{R}^k$ and $Q : [0 : n] \mapsto \mathbb{R}^k$, denoted by $\mathbf{DF}(P, Q)$ is defined as

$$\mathbf{DF}(P,Q) = \min_{\kappa:[1:m+n]\mapsto[0:m],\ \lambda:[1:m+n]\mapsto[0:n]} \max_{t\in[1:m+n]} \left\| P(\kappa(t)) - Q(\lambda(t)) \right\|_2 \qquad (2.2)$$

where κ and λ range over all *discrete*, monotonically increasing, onto mappings of the form $\kappa : [1 : m+n] \mapsto [0 : m]$ and $\lambda : [1 : m+n] \mapsto [0 : n]$.

The discrete Fréchet distance can be explained using the man and dog example as follows. The man and the dog can only stop at vertices of P and Q, and at any step, each of them can either stay at their current vertex or jump to the next one. The discrete Fréchet distance is defined as the minimal leash necessary at these *discrete moments*. In contrast, in case of (continuous) Fréchet distance the man and the dog are not allowed to jump between vertices and must walk through every point on their respective curves.

In case of discrete Fréchet distance, the motion of the dog and the man along the two curves is assumed to be defined by a *monotonic path*. The positions of the man and the dog at any time instant can be represented by a tuple (i, j) if they are at points p_i and q_j on curves P and Q respectively. Unless both the man and the dog have already reached their respective endpoints, at the next time step either both of them should have jumped to the next vertex of their respective curves or one has jumped to the next vertex while the other stays at the same point. Thus the tuple corresponding to their positions at the next time step must be one of the following three: $(i, j+1)$, $(i+1, j)$ or $(i+1, j+1)$. Hence the sequence of (i, j) values at successive time steps forms a *monotonic path*. The discrete Fréchet distance minimizes the required leash length required over all possible monotonic paths that start at (0,0) and end at (m, n).

2.4.2 Computing the Discrete Fréchet distance

The discrete Fréchet distance $\mathbf{DF}(P, Q)$ between two polygonal curves $P \in V^{[0:m]}$ and $Q \in V^{[0:n]}$ can be computed in $O(mn)$ time using algorithm 1 that is based on dynamic programming.

Algorithm 1 Algorithm for computing the discrete Fréchet distance $\mathbf{DF}(P, Q)$

$d_{0,0} = \left\|p_0 - q_0\right\|_2$
for $i = 1$ to m **do**
  $d_{i,0} = \max\left\{d_{i-1,0}\ ,\ \left\|p_i - q_0\right\|_2\right\}$
end for
for $j = 1$ to n **do**
  $d_{0,j} = \max\{d_{0,j-1}\ , \left\|p_0 - q_j\right\|_2\}$
end for
for $i = 1$ to m **do**
  for $j = 1$ to n **do**
    $d_{i,j} = \max\left\{\min\{d_{i,j-1}\ , d_{i-1,j}\ , d_{i-1,j-1}\},\ \left\|p_i - q_j\right\|_2\right\}$
  end for
end for
return $d_{m,n}$

2.4.3 Proof of correctness for algorithm 1

Claim: For all $i = 0, 1, ..., m$ and for all $j = 0, 1, .., n$, $d_{i,j}$ computed by algorithm 1 is equal to $\mathbf{DF}(P|_{[0:i]}, Q|_{[0:j]})$, the discrete Fréchet distance between $P|_{[0:i]}$ and $Q|_{[0:j]}$.

Proof: The proof is by induction on i and j. Consider the base case, $i = j = 0$. Both $P|_{[0:0]} = \langle p_0 \rangle$ and $Q|_{[0:0]} = \langle q_0 \rangle$ consist of a single point each and the only leash length possible is the Euclidean distance between these two points. Thus $\mathbf{DF}(P|_{[0:0]}, Q|_{[0:0]}) = \left\|p_0 - q_0\right\|_2$ and from the algorithm, we see that $d_{0,0} = \mathbf{DF}(P|_{[0:0]}, Q|_{[0:0]})$, proving the claim for the basis.

For the inductive step, assume that $d_{i,j} = \mathbf{DF}(P|_{[0:i]}, Q|_{[0:j]})$ for $i = 0, 2, ..., a - 1$ and $j = 0, 2, ..., b - 1$ for some integers a and b such that $1 \leq a \leq m$ and $1 \leq b \leq n$. We now determine the value of $\mathbf{DF}(P|_{[0:i]}, Q|_{[0:j]})$ for new values of *i* and/or *j* using the man and dog example, assuming that the man and the dog are associated with curves $P|_{[0:i]}$ and $Q|_{[0:i]}$ respectively. There are four possible cases:

- $i = 0,\ j = b$: To find $\mathbf{DF}(P|_{[0:0]}, Q|_{[0:b]})$. The first curve consists of just a single point;

the man remains at this point whereas the dog alone moves forward. It is easy to see that the minimum leash length required for the time period when the dog moves from q_0 to q_b, jumping from one vertex to the next at a time, is the maximum of (i) the leash length required when the dog is at q_b and (ii) the minimum leash length required as the dog moves from q_0 to q_{b-1}. By definition, the second length is $\mathbf{DF}(P|_{[0:0]}, Q|_{[0:b-1]})$ and by the assumption stated earlier, this is equal to $d_{0,b-1}$. Therefore $\mathbf{DF}(P|_{[0:0]}, Q|_{[0:b]}) = \max\{\left\|p_0 - q_b\right\|_2 ,\ d_{0,b-1}\}\ \forall b \in [1:n]$. But from the algorithm we see that this is also equal to $d_{0,b}$ as claimed.

- $i = a,\ j = 0$: To find $\mathbf{DF}(P|_{[0:a]}, Q|_{[0:0]})$. The second curve consists of just a single point; the dog remains at this point whereas the man alone moves forward. It is easy to see that the minimum leash length required for the time period when the man moves from p_0 to p_a, jumping from one vertex to the next at a time, is the maximum of (i) the leash length required when the man is at p_a and (ii) the minimum leash length required as the man moves from p_0 to p_{a-1}. By definition, the second length is $\mathbf{DF}(P|_{[0:a-1]}, Q|_{[0:0]})$ and by the assumption stated earlier, this is equal to $d_{a-1,0}$. Thus $\mathbf{DF}(P|_{[0:a]}, Q|_{[0:0]}) = \max\{\left\|p_a - q_0\right\|_2 ,\ d_{a-1,0}\}\ \forall a \in [1:m]$. But from the algorithm we see that this is also equal to $d_{a,0}$ as claimed.

- $i = a,\ 0 < j < b$: To find $\mathbf{DF}(P|_{[0:a]}, Q|_{[0:j]})$, i.e. the smallest length from among the leash lengths required in various monotonic paths starting at (0,0) and ending at (a, j). The man and the dog are currently at p_a and q_j respectively; neither of them are at their starting points. The monotonic path associated with the motion of the man and the dog currently ends at (a, j) and the path traversed till the previous step would have ended at one of $(a, j-1)$, $(a-1, j)$ or $(a-1, j-1)$. The minimum leash length required for any monotonic path ending at (a, j) is the maximum of (i) the leash length required at the current instant and (ii) the minimum leash length required among monotonic paths

ending at $(a, j-1)$ or $(a-1, j)$ or $(a-1, j-1)$. Hence

$$\mathbf{DF}(P|_{[0:a]}, Q|_{[0:j]}) = \max\left\{ \left\|p_a - q_j\right\|_2 , \min\left\{ \begin{array}{c} \mathbf{DF}(P|_{[0:a]}, Q|_{[0:j-1]}), \\ \mathbf{DF}(P|_{[0:a-1]}, Q|_{[0:j]}), \\ \mathbf{DF}(P|_{[0:a-1]}, Q|_{[0:j-1]}) \end{array} \right\} \right\}$$

By inductive hypothesis, the second term in the $\max$ expression is equivalent to

$$\min\{d_{a,j-1} , d_{a-1,j} , d_{a-1,j-1}\}$$

and hence $\mathbf{DF}(P|_{[0:a]}, Q|_{[0:j]}) = \max\big\{\min\{d_{a,j-1} , d_{a-1,j} , d_{a-1,j-1}\},\ \left\|p_a - q_j\right\|_2\big\}$. This is equal to $d_{a,j}$ as seen from the algorithm and hence the claim holds true in this case too.

- $0 < i < a,\ j = b$: To find $\mathbf{DF}(P|_{[0:i]}, Q|_{[0:b]})$, i.e. the smallest length from among the leash lengths required in various monotonic paths starting at (0,0) ending at (i, b). This can be proved to be equal to $d_{i,b}$ proving our claim for this case too. The proof is similar to that for the previous case.

Thus, by the principle of mathematical induction we have proved that $d_{i,j} = \mathbf{DF}(P|_{[0:i]}, Q|_{[0:j]})$ for $i = 0, 1, ..., m$ and $j = 0, 1, ..., n$. It follows that $d_{m,n} = \mathbf{DF}(P|_{[0:m]}, Q|_{[0:n]}) = \mathbf{DF}(P, Q)$ since $P \equiv P|_{[0:m]}$ and $Q \equiv Q|_{[0:n]}$ and hence algorithm 1 is correct.

2.4.4 Matching under transformation of curves

A more general scenario in curve matching is finding similarity between curves in the presence of transformations, such as scaling, rotation, translation or some combination of these. This can be posed as a *decision problem*: given a group G of admissible transformations, the aim is to determine if there is some transformation $g \in G$ which when applied on a curve Q makes it closely resemble another curve P, i.e $d(P, gQ) \leq \epsilon$, where gQ is the curve obtained by applying g on Q, d is the distance measure for finding curve similarity and ϵ is a threshold for deciding whether the curves match.

Algorithm 2 Matching with discrete Fréchet distance under scaling in 2-dimensions

$C = \emptyset$
for $i = 0$ to m **do**
  for $j = 0$ to n **do**
    solve the equation $\left\|p_i - \lambda q_j\right\|_2^2 - \epsilon^2 = 0$ to get λ
    if solution exists for the above equation **then**
      $C = C \cup \{\lambda\}$ (use any one of the 2 solutions for λ)
    end if
  end for
end for
for each $\lambda \in C$ **do**
  $Q' = \langle \lambda q_0, \lambda q_1, ..., \lambda q_n \rangle$
  if $\mathbf{DF}(P, Q') \leq \epsilon$ **then**
    return λ
  end if
end for
return false

The above scenario may arise in case of search in handwritten documents. Words in a handwritten document may have been written in different orientations. Different instances of the same word may have been written in different sizes. When the user writes the query, it may not be of the same size or orientation as the matching words in the document. To be able to retrieve the matches under such conditions, the search algorithm must match curves considering scaling and rotation as being admissible transformations. Algorithm 1 cannot be used here since it does not consider transformations.

Mosig and Clausen [8] propose an efficient algorithm for solving the above mentioned decision problem for polygonal curves with the distance measure being discrete Fréchet distanceand G being a subgroup of affine transformations. This algorithm can handle scaling and rotation about the origin. We present only a very brief overview of their method here. Their

algorithm is based on the idea of identifying the *equivalence classes* associated with a particular *equivalence relation* on *transporter sets*, which are subsets of G. These equivalence classes allow the algorithm to solve the decision problem by *considering only a finite number of transformations* even though the group G is *typically an infinite set*. The set of transformations that are considered by the algorithm is called a *suptransversal*. The number of transformations in the *suptransversal* is $O(mn)$ and the time complexity of the algorithm is $O(m^2n^2)$, where m and n are the lengths of the two polygonal curves.

Algorithm 3 Matching with discrete Fréchet distance under rotation about the origin in 2-D

$C = \emptyset$
for $j = 0$ to n **do**
  $r = (q_j^x \cos\theta - q_j^y \sin\theta,\ q_j^x \sin\theta + q_j^y \cos\theta)$
  for $i = 0$ to m **do**
    solve the equation $\left\|p_i - r\right\|_2 - \epsilon^2 = 0$ to get θ
    if solution exists for the above equation **then**
      $C = C \cup \{\theta\}$ (use any one of the 2 solutions for θ)
    end if
  end for
end for
for each $\theta \in C$ **do**

$$A_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

  $Q' = \langle A_\theta q_0, A_\theta q_1, ..., A_\theta q_n \rangle$
  if $\mathbf{DF}(P, Q') \leq \epsilon$ **then**
    return θ
  end if
end for
return false

Algorithms 2 solves the decision problem of determining if $\mathbf{DF}(P, gQ) \leq \epsilon$ for g belonging to the group of scaling about the origin in a 2-dimensional space. Algorithm 3 solves a similar problem but for the group of rotations about the origin in 2-dimensions. P and Q are polygonal curves in a 2-dimensional space and are of length m and n respectively. The algorithms compute the *suptransversals* in terms of certain ranges of scalar parameter values - the parameter being the scaling factor in case of algorithm 2 and the rotation angle in case of algorithm 3. The algorithms choose one value (λ in the former algorithm, θ in the latter) from each range. λq_j in the algorithm denotes the point whose coordinate values are the obtained by scaling the corresponding coordinate values of point q_j by the scalar value λ. Similarly $A_\theta p_i$ denotes the point corresponding to the vector obtained as the product of matrix A_θ and the vector denoting the point q_j. If there exists any valid transformation g for which $\mathbf{DF}(P, gQ) \leq \epsilon$, then the algorithms return the parameter value associated with one such transformation. If there is no such transformation, then the algorithms return *false*.

2.5 Partial Fréchet distance

When the search query is a *subword* rather than a whole word, the search algorithm should test whether any contiguous portion (substring) of any sample word (approximately) matches the query. Essentially the search algorithm must check whether any *subcurve* of a *sample curve* approximately matches the entire *query curve*. Discrete Fréchet distance is not suitable for such *subcurve matching*. Mosig and Clausen [8] propose a variant of the discrete Fréchet distance called the *partial Fréchet distance* for measuring the resemblance of one curve as a subcurve of another.

Formally, the *partial Fréchet distance* $\mathbf{PF}(P, Q)$ between two polygonal curves $P \in V^{[0:m]}$ and $Q \in V^{[0:n]}$ is defined as

$$\mathbf{PF}(P, Q) = \min_{[a:b] \subseteq [0:m]} \mathbf{PF}(P|_{[a:b]}, Q) = \min_{0 \leq a \leq b \leq m} \mathbf{PF}(P|_{[a:b]}, Q) \tag{2.3}$$

respectively, can be computed in $O(mn)$ time using algorithm 4 that is based on dynamic programming. In terms of the man-dog example, the partial Fréchet distance minimizes the

required leash length required over all possible monotonic paths that start at $(a, 0)$ and end at (b, n) where a and b are any two integers such that $0 \leq a \leq b \leq m$. Thus the man can start at any vertex of P, not just p_0 and can stop at starting point itself or at any vertex of P that occurs after the starting point. The dog, however, must move from q_0 to q_n. The other restrictions on the movements of the man and the dog are the same as for discrete Fréchet distance and here too their motion is associated with a *monotonic path*.

2.5.1 Proof of correctness for algorithm 4

The following proof is based on the idea that computing $\mathbf{PF}(P, Q)$ is inherently equivalent to finding a monotonic path ranging from $(a, 0)$ to (b, n) for any two integers a and b, $0 \leq a \leq b \leq m$, such that the discrete Fréchet distance associated with this path is the minimum among all monotonic paths of the same form. Note that unlike in case of discrete Fréchet distance, *the monotonic paths considered here can start at* $(a, 0)$ *for any* $a \in [0 : m]$, *not just* (0,0). As before, assume that the man and the dog move along $P|_{[0:i]}$ and $Q|_{[0:j]}$ respectively.

Claim: For all $i = 0, 1, ..., m$ and for all $j = 0, 1, .., n$, $d_{i,j}$ computed by algorithm 4 is equal to $\min_{0 \leq c \leq i} \mathbf{DF}(P|_{[c:i]}, Q|_{[0:j]})$, the minimum discrete Fréchet distance between any subcurve of P ending at p_i and the (sub)curve $Q|_{[0:j]}$.

Proof: The proof is by induction on i and j. Consider the base case, $i \in [0 : m], j = 0$. Here $Q|_{[0:j]} = Q|_{[0:0]}$, i.e. it consists of just the single point q_0. Observe that

$$\mathbf{DF}(P|_{[c:i]}, Q|_{[0:0]}) \geq \left\|p_i - q_0\right\|_2 \; \forall c \in [0 : i]$$

since the discrete Fréchet distance between any subcurve of the form $P|_{[c:i]}$ and the point q_0 cannot be less than the leash length required when the man and the dog are at their respective endpoints p_i and q_0 respectively. This length is the same as $\mathbf{DF}(P|_{[i:i]}, Q|_{[0:0]})$ and hence

$$\left\|p_i - q_0\right\|_2 = \min_{c \in [0:i]} \mathbf{DF}(P|_{[c:i]}, Q|_{[0:0]})$$

Algorithm 4 Algorithm for computing the partial Fréchet distance $\mathbf{PF}(P, Q)$

for $i = 0$ to m **do**
  $d_{i,0} = \left\|p_i - q_0\right\|_2$
  $a_{i,0} = i$
end for
for $j = 1$ to n **do**
  $d_{0,j} = \max\left\{d_{0,j-1}\,, \left\|p_0 - q_j\right\|_2\right\}\ ;\ a_{0,j} = 0$
end for
for $i = 1$ to m **do**
  for $j = 1$ to n **do**
    $d_{i,j} = \max\left\{\min\{d_{i,j-1}\,, d_{i-1,j}\,, d_{i-1,j-1}\},\ \left\|p_i - q_j\right\|_2\right\}$
    if $d_{i,j} = d_{i,j-1}$ **then**
      $a_{i,j} = a_{i,j-1}$
    else if $d_{i,j} = d_{i-1,j}$ **then**
      $a_{i,j} = a_{i-1,j}$
    else if $d_{i,j} = d_{i-1,j-1}$ **then**
      $a_{i,j} = a_{i-1,j-1}$
    else
      $a_{i,j} = \min\left\{a_{i,j-1}\ ,\ a_{i-1,j}\ ,\ a_{i-1,j-1}\right\}$
    end if
  end for
end for
$minlen = -\infty$
for $i = 0$ to m **do**
  if $minlen > d_{i,n}$ **then**
    $minlen = d_{i,n}\ ;\ lastidx = i$
  end if
end for
return $minlen$ $\{P|_{[a_{lastidx,n}:lastidx]}$ is the closest matching subcurve$\}$

Since algorithm 4 sets $d_{i,0}$ to $\left\|p_i - q_0\right\|_2$ for $i = 0, 1, ..., m$, the claim is proved for the base case.

Inductive hypothesis: Assume that $d_{i,j} = \min_{0 \leq c \leq i} \mathbf{DF}(P|_{[c:i]}, Q|_{[0:j]})$ for $i = 0, 2, ..., a-1$ and $j = 0, 2, ..., b-1$ for some integers a and b such that $1 \leq a \leq m$ and $1 \leq b \leq n$. We now determine the value of $\mathbf{DF}(P|_{[0:i]}, Q|_{[0:j]})$ for new values of *i* and/or *j* using the man and dog example.

For the inductive step we consider three possible cases:

- $i = 0,\ j = b$: To find $\min_{[c:0] \subseteq [0:0]} \mathbf{DF}(P|_{[c:0]}, Q|_{[0:b]})$. This equals $\mathbf{DF}(P|_{[0:0]}, Q|_{[0:b]})$ since c can only be 0 in this case. The man is currently at his starting point but the dog is not at its starting point; hence this is not the first time step. The monotonic path covered till the current time step ends at $(0, b)$ and the path traversed till the previous step must have ended at $(0, b-1)$. Note that there can be only one monotonic path ending at $(0, j)$ for any (valid) value of j. The minimum leash length required for any monotonic path ending at $(0, j)$ is the maximum of (i) the leash length required at the current instant and (ii) the minimum leash length required for the monotonic path ending at $(0, b-1)$. By definition, the latter is equivalent to $\mathbf{DF}(P|_{[0:0]}, Q|_{[0:b-1]})$ and by inductive hypothesis this value equals $d_{0,b-1}$. Hence $\mathbf{DF}(P|_{[0:0]}, Q|_{[0:b]}) = \max\{d_{0,b-1},\ \left\|p_0 - q_b\right\|_2\}$. It is now easy to see that the algorithm sets $d_{0,b}$ to $\mathbf{DF}(P|_{[0:0]}, Q|_{[0:b]})$. Thus the claim is true in this case.

- $0 < i < a,\ j = b$: To find $\min_{0 \leq c \leq i} \mathbf{DF}(P|_{[c:i]}, Q|_{[0:b]})$. Since the dog is not at its starting point this is not the first time step. The minimum leash length required for any monotonic path ending at (i, j) is the maximum of (i) the leash length required at the current instant and (ii) the minimum leash length required among monotonic paths

ending at one of $(i, b-1)$, $(i-1, b)$ or $(i-1, b-1)$. Hence by definition

$$\min_{0\leq c\leq i} \mathbf{DF}(P|_{[c:i]}, Q|_{[0:b]}) = \max \left\{ \begin{array}{c} \left\|p_i - q_b\right\|_2 \\ \min \left\{ \begin{array}{c} \min_{0\leq c\leq i} \mathbf{DF}(P|_{[c:i]}, Q|_{[0:b-1]}), \\ \min_{0\leq c\leq (i-1)} \mathbf{DF}(P|_{[c:i-1]}, Q|_{[0:b]}), \\ \min_{0\leq c\leq (i-1)} \mathbf{DF}(P|_{[c:i-1]}, Q|_{[0:b-1]}) \end{array} \right\} \end{array} \right\}$$

By inductive hypothesis the second term in the $\max$ expression is equivalent to

$$\min\{d_{i,b-1}\ , d_{i-1,b}\ , d_{i-1,b-1}\}$$

and hence

$$\min_{0\leq c\leq i} \mathbf{DF}(P|_{[c:i]}, Q|_{[0:b]}) = \max\left\{ \left\|p_i - q_b\right\|_2\ ,\ \min\left\{d_{i,b-1}\ , d_{i-1,b}\ , d_{i-1,b-1}\right\}\right\}$$

It is now easy to see that the algorithm sets $d_{i,b}$ to $\min_{0\leq c\leq i} \mathbf{DF}(P|_{[c:i]}, Q|_{[0:b]})$. Thus the claim is true in this case.

- $i = a,\ 0 < j < b$: To find $\min_{0\leq c\leq a} \mathbf{DF}(P|_{[c:a]}, Q|_{[0:j]})$. Since $j > 0$, the dog is not at its starting point and hence this is not the first time step. The monotonic path traversed till the previous time step must end at one of $(a, j-1)$, $(a-1, j)$ or $(a-1, j-1)$. Proceeding in the same way as in the previous case, it can be proved that the claim holds true for this case as well.

Thus using mathematical induction we have proved that $d_{i,j} = \min_{0\leq c\leq i} \mathbf{DF}(P|_{[c:i]}, Q|_{[0:j]})$ for $i = 0, 1, ..., m$ and $j = 0, 1, ..., n$. Using the fact that $Q \equiv Q|_{[0:n]}$, we get

$$d_{i,n} = \min_{0\leq c\leq i} \mathbf{DF}(P|_{[c:i]}, Q) \text{ for } i = 0, 1, ..., m \tag{2.4}$$

The value of $minlen$ returned by the algorithm is equal to $\min_{i\in[0:m]} d_{i,n}$. Using equation (2.4) and denoting by $return\_value$ the value that is returned, we get

$$\begin{aligned} return\_value &= \min_{i\in[0:m]} \min_{0\le c\le i} \mathbf{DF}(P|_{[c:i]}, Q) \\ &= \min_{0\le c\le i\le m} \mathbf{DF}(P|_{[c:i]}, Q) = \min_{[c:i]\subseteq[0:m]} \mathbf{DF}(P|_{[c:i]}, Q) \\ &= \mathbf{PF}(P, Q) \qquad (2.5) \\ &(\text{by definition of } \mathbf{PF}(P, Q)) \end{aligned}$$

Thus algorithm 4 is correct as it returns the partial Fréchet distance between P and Q as expected.

2.5.2 The Partial Dynamic Time Warping Distance (PDTW)

Discrete Fréchet distance is closely related to the well known DTW distance [8]. The discrete Fréchet distance formulation in equation (2.2) is equivalent to the formulation

$$\mathbf{DF}(P, Q) = \min_{\kappa:[1:m+n]\mapsto[0:m],\ \lambda:[1:m+n]\mapsto[0:n]} \left\| P\circ\kappa - Q\circ\lambda \right\|_\infty \qquad (2.6)$$

where κ and λ range over all *discrete*, monotonically increasing, onto mappings of the form $\kappa : [1 : m + n] \mapsto [0 : m]$ and $\lambda : [1 : m + n] \mapsto [0 : n]$. This formulation is based on the definition of the L_∞-norm for vector sequences, given in section 2.2. On the other hand, the DTW distance between polygonal curves P and Q is defined as

$$\mathbf{DTW}(P, Q) = \min_{\kappa:[0:K]\mapsto[0:m],\ \lambda:[0:K]\mapsto[0:n]} \left\| P\circ\kappa - Q\circ\lambda \right\|_2 \qquad (2.7)$$

where κ and λ range over all *discrete*, monotonically increasing, onto mappings of the form $\kappa : [0 : K] \mapsto [0 : m]$ and $\lambda : [0 : K] \mapsto [0 : n]\ \ \forall K \in [\max(m, n), m + n]$.

For the purpose of benchmarking partial Fréchet distance, we define the *partial dynamic*

time warping distance analogously as:

$$\mathbf{PDTW}(P, Q) = \min_{[a:b] \subseteq [0:m]} \mathbf{DTW}(P|_{[a:b]}, Q)$$

It can be computed using the recursions used in case of partial Fréchet distance, with the difference being that the max operation for finding L_∞-norm is replaced everywhere by addition of squares to find L_2-norm instead.

2.6 Partial search using Fréchet and DTW variants

We propose new variants of the partial Fréchet distance (PFD) and the partial DTW (PDTW) and give dynamic programming algorithms to compute these. The need for these new variants arises since PFD and PDTW are not well-suited for the type of partial curve matching that is needed for partial search in handwriting. For search in handwriting, we need to capture *only the similarity in the shape* of curves; PFD and PDTW do not always capture the shape similarity. We discuss this in detail below.

2.7 Need for translation-invariant matching

Discrete Fréchet distance and partial Fréchet distance are not translation invariant in general. The discrete Fréchet distance between a curve and a translated version of the same curve is not 0 but is equal to the distance between the starting points of the original curve and translated version, even though their shapes are exactly same. Thus when the curves being matched are the polygonal curves formed by the (x, y) coordinates of handwritten text, discrete Fréchet distance and partial Fréchet distance do not capture fully the idea of shape similarity that we need for search in handwriting. When searching for a handwritten pattern, we must eliminate the effect of translation since the aim is to compare only the shapes of the query and a sample word. We should be able to find a matching word irrespective of the position within the word where the query occurs as a substring and the position of the word in a page of handwritten text.

Since handwritten text has only x and y coordinates as its spatial features, we restrict ourselves to matching of curves in two-dimensional space. For this setting, we propose two modified versions of the partial Fréchet distance for performing translation-invariant partial curve matching. The first version operates on the same (x, y) feature space whereas the second version operates on a (y, θ) feature space, where y now denotes the *height* feature that is computed using only the y-coordinates as explained in section 2.9, and θ denotes the angle between the x-axis and the the slope of the tangent to the original curve at any sample point. In the following sections we assume that P and Q are the original polygonal curves in (x, y) dimensions that are to be matched (partial matching). $P = \langle p_0, p_1, ..., p_m \rangle,\ p_i \in \mathbb{R}^2\ \forall i = 0, 1, ..., m$ and $Q = \langle q_0, q_1, ..., q_n \rangle,\ q_j \in \mathbb{R}^2\ \forall j = 0, 1, ..., n$.

Since PDTW too is not translation invariant, and since it is closely related to PFD, we define new variants of PDTW that are analogous to those of PFD.

2.8 A variant operating on the (x, y) feature space

We denote by $\mathbf{PFD_{x,y}}$ the first variant of the partial Fréchet distance that operates on the (x, y) feature space. We formulate this variant by considering the following. For any subcurve C of P that needs to be checked for resemblance in shape with Q, the effect of translation can be eliminated by shifting both C and Q such that both have the same starting point, say the origin $(0, 0)$. The shape similarity between the C and Q can then found using the discrete Fréchet distance between the shifted (sub)curves. But this method is computationally very expensive. The number of distinct subcurves of P is $O(m^2)$ and the total time for computing finding all discrete Fréchet distances is $O(m^3 n)$. To avoid this problem we make a heuristic assumption that reduces the number of subcurves that are checked for a potential match with Q. We assume that if the subcurve of P that matches with $Q|_{[0:c]}$ $(n \geq c > 0)$ most closely in shape is $P|_{[a:b]}$ $(m \geq b > a \geq 0)$, then the subcurve of P that matches with $Q|_{[0:c-1]}$ most closely in shape is one of the at most three possible subcurves - $P|_{[a-1:b-1]}$, $P|_{[a-1:b]}$ or $P|_{[a:b-1]}$.

The analogous variant of PDTW: Analogous to the $\mathbf{PFD_{x,y}}$ variant of PFD, we define the $\mathbf{PDTW_{x,y}}$ variant of the partial DTW. It is based on the same idea of shifting the subcurve

and Q to make their starting points coincide with (0,0), but instead of discrete Fréchet distance, DTW is used to find the similarity between the shifted (sub)curves. It can be computed in $O(mn)$ time using the algorithm for $\mathbf{PFD_{x,y}}$ with the max operation replaced by squaring and addition, so as to find L_∞-norm instead of L_2-norm.

2.8.1 Computation of $\mathbf{PFD_{x,y}}$

We present a DP algorithm that computes $\mathbf{PFD_{x,y}}(P, Q)$ in $O(mn)$ time. The algorithm essentially performs the following recurrence equations:

$$
\begin{aligned}
&init_{i,0} = i,\ \forall i \in [0:m] \\
&init_{0,j} = 0,\ \forall j \in [1:n] \\
&d_{i,0} = 0,\ \forall i \in [0:m] \\
&d_{0,j} = \max\left\{d_{0,j-1},\ \left\|a_0 - b_j\right\|_2\right\},\ \forall j \in [1:n] \\
&\Delta_{i,j} = q_0 - p_{init_{i,j}},\ \forall i \in [0:m],\ \forall j \in [0:n] \\
&t_{i,j} = p_i - q_j,\ \forall i \in [0:m],\ \forall j \in [0:n] \\
&d_{i,j} = \min\left\{\begin{array}{l} \max\left\{d_{i-1,j},\ \left\|t_{i,j} + \Delta_{i-1,j}\right\|_2\right\}, \\ \max\left\{d_{i,j-1},\ \left\|t_{i,j} + \Delta_{i,j-1}\right\|_2\right\}, \\ \max\left\{d_{i-1,j-1},\ \left\|t_{i,j} + \Delta_{i-1,j-1}\right\|_2\right\} \end{array}\right\} \\
&init_{i,j} = \begin{cases} init_{i-1,j}, \text{ if } d_{i,j} = d_{i-1,j} \\ init_{i,j-1}, \text{ if } d_{i,j} \neq d_{i-1,j}\ \&\ d_{i,j} = d_{i,j-1} \\ init_{i-1,j-1}, \text{ otherwise} \end{cases}
\end{aligned}
$$

At the end of the above recursion, $\mathbf{PFD}_{\mathbf{x},\mathbf{y}}(P,Q)$ and the subcurve $P|_{[a:b]}$ of P that is the closest match to Q according to $\mathbf{PFD}_{\mathbf{x},\mathbf{y}}$ are found using the formulae

$$\mathbf{PFD}_{\mathbf{x},\mathbf{y}}(P,Q) = \min_{i\in[0:m]} d_{i,n} \text{ , where}$$

$$b = \text{arg}\min_{i\in[0:m]} d_{i,n}$$

$$a = init_{b,n}$$

2.9 A variant operating on the (y,θ) feature space

We denote by $\mathbf{PFD}_{\mathbf{y},\theta}$ the variant of the partial Fréchet distance that operates on the (y,θ) feature space. Here y denotes the height of a sample point from the horizontal base of the curve, i.e. the difference between the y-coordinates values of a given sample point and the sample point of the curve having the smallest y-coordinate value (the *lowest* sample point). θ is the counterclockwise angle between the x-axis and the slope of the tangent to the curve at a given sample point. We formulate this variant based on the following observations:

- θ is translation-invariant since its value at a particular sample point of the curve does not vary even if the curve is shifted along the horizontal and/or the vertical axes.
- The height feature y is also translation invariant since it is the vertical distance *relative* to the lowest sample point, which will be the lowest point even if the curve is shifted along either of the axes.
- It is easy to observe that y is also the y-coordinate feature of a curve obtained by shifting the original curve such that its lowest point now has the y-coordinate value 0. Thus using the height feature is essentially equivalent to shifting all the curves along the y-axis so that all have the same lowest y-coordinate value, 0, thereby nearly eliminating the effect of translation on the y-coordinate feature.
- It is not meaningful to use the previous idea for eliminating the effect of translation on x-coordinate feature. Consider for instance a query that matches with a substring

occurring in the middle of a whole word. Now if both the query curve and the sample curve are shifted so that their leftmost points have the same x-coordinate value, say 0, then we lose the ability to check whether the query matches with some middle portion of the sample word since the matching subcurve is shifted away from the query curve along the x-axis and PFD as well as PDTW will fail to capture the shape similarity.

- It is easy to see that given only the (y, θ) values corresponding to all the sample points of a curve, it is possible to reconstruct the shape of the original curve. Since we are interested only in matching the shape of curves rather than the curves themselves, it appears that y and θ features alone capture all the required details about the curves to be matched.

The $\mathbf{PFD}_{\mathbf{y},\theta}$ variant is equivalent to transforming the curves which are originally in the (x, y) space to the (y, θ) feature space and then computing the actual partial Fréchet distance between the transformed curves. Since (y, θ) is translation-invariant, the effect of translation in (x, y) space does not carry over to the (y, θ) feature space. Hence PFD can be used directly on the transformed curves.

It should be noted that the original partial Fréchet distance is based on the Euclidean distance between matched points. While the Euclidean distance between two points in (x, y) feature space is a very intuitive measure of finding how close the points are, it is less intuitive when it is applied to points in the transformed space, here the (y, θ) feature space. Unlike x and y which are analogous features from the 2-D space in which the original curves lie, y and θ are very different from each other. The relative importance of ty and θ in determining the closeness of two points for the purpose of matching them is not obvious. To mitigate this problem, we use the *weighted Euclidean distance* in finding the partial Fréchet distance between the transformed curves and determine a good set of weights empirically. We denote by $\mathbf{wPF}(P, Q, w_1, w_2)$ the modified partial Fréchet distance between curves P and Q that uses *weighted Euclidean distance* with weights w_1 and w_2 instead of the original L_2-norm Euclidean distance. The algorithm to compute $\mathbf{wPF}(P, Q, w_1, w_2)$ is essentially the same as algorithm 4, but the weighted Euclidean distance, with weights w_1 and w_2 respectively, is used instead of the L_2-norm, $\left\|.\right\|_2$.

The analogous variant of PDTW: Analogous to the $\mathbf{PFD}_{\mathbf{y},\theta}$ variant of PFD, we define the $\mathbf{PDTW}_{\mathbf{y},\theta}$ variant of the partial DTW. It is based on the same idea of transforming the curves to (y, θ) space and computing the distance between the transformed curves, but instead of PFD, PDTW is used as the distance measure between the transformed curves. It can be computed in $O(mn)$ time using an algorithm similar to that for partial Fréchet distance, with the only difference being that the max operator is to be replaced everywhere by squaring and addition, so as to find L_∞-norm instead of L_2-norm.

Algorithm 5 Algorithm to compute $\mathbf{PFD}_{\mathbf{y},\theta}(P, Q)$

$ymin_P = p_0^y$

for $i = 1$ to m **do**

$ymin_P = \min\{p_i^y, ymin_P\}$

end for

$ymin_Q = q_0^y$

for $j = 1$ to n **do**

$ymin_Q = \min\{q_j^y, ymin_Q\}$

end for

for $i = 0$ to m **do**

$a_i^y = p_i^y - ymin_P$

end for

for $j = 0$ to n **do**

$b_j^y = q_j^y - ymin_Q$

end for

$a_0^\theta = \tan^{-1} \frac{p_y^1 - p_y^0}{p_x^1 - p_x^0}$

$a_m^\theta = \tan^{-1} \frac{p_y^m - p_y^{m-1}}{p_x^m - p_x^{m-1}}$

for $i = 1$ to $m - 1$ **do**

$a_i^\theta = 0.5 * \left(\tan^{-1} \frac{p_y^{i+1} - p_y^i}{p_x^{i+1} - p_x^i} + \tan^{-1} \frac{p_y^i - p_y^{i-1}}{p_x^i - p_x^{i-1}}\right)$

end for

$b_0^\theta := \tan^{-1} \frac{q_y^1 - q_y^0}{q_x^1 - q_x^0}$

$b_n^\theta := \tan^{-1} \frac{q_y^n - q_y^{n-1}}{q_x^n - q_x^{n-1}}$

for $j = 1$ to $n - 1$ **do**

$b_j^\theta := 0.5 * \left(\tan^{-1} \frac{q_y^{j+1} - q_y^j}{q_x^{j+1} - q_x^j} + \tan^{-1} \frac{q_y^j - q_y^{j-1}}{q_x^j - q_x^{j-1}}\right)$

end for

$A = \langle a_0, a_1, \ldots, a_m \rangle$

$B = \langle b_0, b_1, \ldots, b_n \rangle$

return $\mathbf{wPF}(A, B, w_1, w_2)$ $\{w_1$ and w_2 determined empirically by the user$\}$

2.9.1 Computation of $\mathbf{PFD}_{\mathbf{y},\theta}$

Algorithm 5 computes $\mathbf{PFD}_{\mathbf{y},\theta}(P, Q)$. The x, y and θ feature values of any point $c \in \mathbb{R}^2$ are denoted by c^x, c^y and c^θ respectively. $\mathbf{PF}(A, B)$ is computed using the algorithm in section 2.5, treating A and B as if they are curves in the (x, y) feature space. Thus the y and θ values of the A and B curve are treated as being their x, y coordinate values. Transforming the curves to the new feature space takes only $O(m + n)$ time as seen from the above algorithm. The resultant curves have the same number of sample points as the original curves and hence the time for computing $\mathbf{PF}(A, B)$ is $O(mn)$. Thus the total time for computing $\mathbf{PFD}_{\mathbf{y},\theta}(A, B)$ is $O(mn)$.

Chapter 3

A Pair HMM for subcurve matching

PHMMs generate two sequences simultaneously and provide a joint probability distribution over finite length sequences [12]. They are typically used in bioinformatics as generative models of local and global alignments of sequences, used to find similarity between biological sequences. We formulate a PHMM that works on polygonal curves rather on biological sequences and provides a probabilistic interpretation of partial curve matching.

3.1 The proposed model

Our model, shown in figure 3.1, provides a joint probability distribution over alignments between pairs of polygonal curves of finite length. In particular, we want to determine the probability that a curve Q matches approximately with a particular subcurve of another curve of P via a particular curve alignment. States **M**, **P** and **Q** model the alignment between Q and the matching subcurve of P, and can be explained intuitively in terms of the man-dog analogy. Assume that the dog walks along curve Q and that the man walks along the corresponding matching subcurve of P. State **M** corresponds to time instances when both the man and the dog move forward to the next vertex of their respective curves. State **P** corresponds to instances when only the man moves forward to the next point on P while the dog stationary, and state **Q** corresponds to vice-versa. States **P'** and **P"** respectively generate the initial and terminating portions of P that are not aligned with Q.

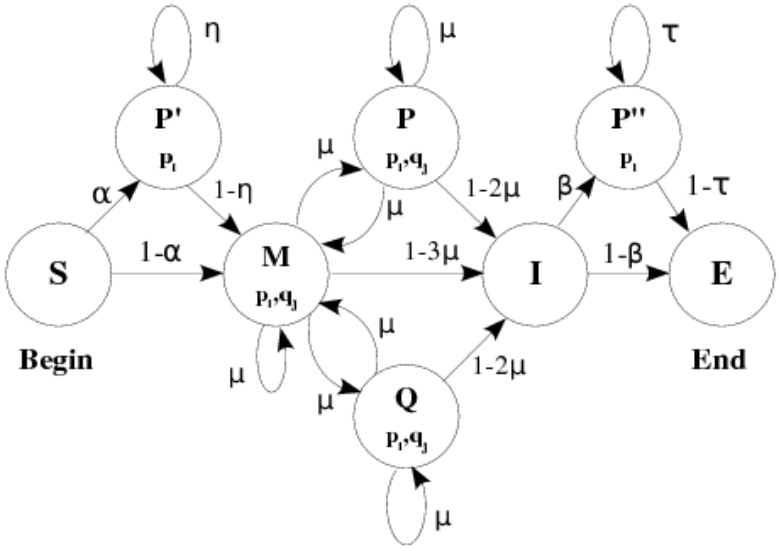

Figure 3.1: A Pair HMM for partial matching of curves

3.2 Determining the most probable alignment

We assume that the polygonal curves P and Q are in the (y, θ) feature space. This avoids the effect of translation on partial matching and there is no need for shifting of curves while computing the probability score of partial matching. Using the Viterbi algorithm, we can find the most probable alignment of Q as a subcurve of P as well as the probability associated with this alignment. We use the following recursions to find this probability:

$$w_{P'}(i,0) = \alpha\eta^i, \forall i \geq 0 \tag{3.1}$$

$$w_{P'}(i,j) = 0, \forall j \geq 1 \tag{3.2}$$

$$w_M(0,0) = (1-\alpha)\, g(p_0, q_0) \tag{3.3}$$

$$w_M(0,j) = 0, \forall j \geq 1 \tag{3.4}$$

$$w_M(i,0) = (1-\eta)\, g(p_i, q_0)\, w_{P'}(i-1,0), \forall i \geq 1 \tag{3.5}$$

$$w_P(0,j) = 0, \forall j \geq 0 \tag{3.6}$$

$$w_Q(i,0) = 0, \forall i \geq 0 \tag{3.7}$$

$$w_{P''}(0,j) = 0, \forall j \geq 0 \tag{3.8}$$

$$w_{M}(i,j) = \mu\, g(p_i, q_j)\ \max \begin{cases} w_{M}(i-1,j-1) \\ w_{P}(i-1,j-1) \\ w_{Q}(i-1,j-1) \end{cases} , \forall i,j \geq 1 \tag{3.9}$$

$$w_{P}(i,j) = \mu\, g(p_i, q_j)\ \max \begin{cases} w_{M}(i-1,j) \\ w_{P}(i-1,j) \end{cases} , \forall i \geq 1\, \forall j \geq 0 \tag{3.10}$$

$$w_{Q}(i,j) = \mu\, g(p_i, q_j)\ \max \begin{cases} w_{M}(i,j-1) \\ w_{Q}(i,j-1) \end{cases} , \forall i \geq 0\, \forall j \geq 1 \tag{3.11}$$

$$w_{I}(i,j) = \max \begin{cases} (1-3\mu)\, w_{M}(i,j) \\ (1-2\mu)\, w_{P}(i,j) \\ (1-2\mu)\, w_{Q}(i,j) \end{cases} , \forall i \geq 0\, \forall j \geq 0 \tag{3.12}$$

$$w_{P''}(i,j) = \max \begin{cases} \beta\, w_{I}(i-1,j) \\ \tau\, w_{P''}(i-1,j) \end{cases} , \forall i \geq 1\, \forall j \geq 0 \tag{3.13}$$

$$w_{E}(i,j) = \max \begin{cases} (1-\beta)\, w_{I}(i,j) \\ (1-\tau)\, w_{P''}(i,j) \end{cases} , \forall i \geq 0\, \forall j \geq 0 \tag{3.14}$$

In these equations, $g(p_i, q_j)$ denotes the emission probability of the pair (p_i, q_j). Since p_i and $q_j)$ can be any two points in a two-dimensional Euclidean space having infinite number of points, the emission probability that we use is actually a probability density function and not a probability value. We want $g(p_i, q_j)$ to be a decreasing function of the distance between the points p_i and q_j. We chose $g(p_., q_.)$ to be the exponential probability density function:

$$g(p_i, q_j) = \gamma e^{-\gamma D_w(p_i, q_j)},\ \text{where}$$
$$D_w = w * (p_i^y - q_j^y)^2 + (1-w) * (p_i^\theta - q_j^\theta)^2$$

D_w is the squared weighted Euclidean distance. The Viterbi alignment, or the alignment with the highest probabilistic score is the alignment corresponding to $w_{E}(m,n)$. To find the subcurve of curve P to which curve Q is aligned in this Viterbi alignment, we need to find only the starting and ending points of that subcurve. These can be obtained using another set of

recursions that track the alignments chosen by the Viterbi algorithm.

$$r_{M}(i,0) = i, \forall i \geq 0 \tag{3.15}$$

$$r_{M}(0,j) = -1, \forall j \geq 1 \tag{3.16}$$

$$r_{P}(0,j) = -1, \forall j \geq 0 \tag{3.17}$$

$$r_{Q}(i,0) = -1, \forall i \geq 0 \tag{3.18}$$

$$r_{P''}(0,j) = -1, \forall j \geq 0 \tag{3.19}$$

$$t_{P''}(0,j) = -1, \forall j \geq 0 \tag{3.20}$$

$$u_{M}(i,j) = \max\left\{w_{M}(i-1,j-1), w_{P}(i-1,j-1), w_{Q}(i-1,j-1)\right\}, \forall i,j \geq 1 \tag{3.21}$$

$$r_{M}(i,j) = \begin{cases} r_{M}(i-1,j-1), \text{if } u_{M}(i,j) = w_{M}(i-1,j-1) \\ r_{P}(i-1,j-1), \text{if } u_{M}(i,j) = w_{P}(i-1,j-1) \\ r_{Q}(i-1,j-1), \text{if } u_{M}(i,j) = w_{Q}(i-1,j-1) \end{cases}, \forall i,j \geq 1 \tag{3.22}$$

$$u_{P}(i,j) = \max\left\{w_{M}(i-1,j), w_{P}(i-1,j)\right\}, \forall i \geq 1 \, \forall j \geq 0 \tag{3.23}$$

$$r_{P}(i,j) = \begin{cases} r_{M}(i-1,j), \text{if } u_{P}(i,j) = w_{M}(i-1,j) \\ r_{P}(i-1,j), \text{if } u_{P}(i,j) = w_{P}(i-1,j) \end{cases}, \forall i,j \geq 1 \tag{3.24}$$

$$u_{Q}(i,j) = \max\left\{w_{M}(i,j-1), w_{Q}(i,j-1)\right\}, \forall i \geq 0 \, \forall j \geq 1 \tag{3.25}$$

$$r_{Q}(i,j) = \begin{cases} r_{M}(i,j-1), \text{ if } u_{Q}(i,j) = w_{M}(i,j-1) \\ r_{Q}(i,j-1), \text{ if } u_{Q}(i,j) = w_{Q}(i,j-1) \end{cases}, \forall i,j \geq 1 \tag{3.26}$$

$$r_{I}(i,j) = \begin{cases} r_{M}(i,j), \text{if } w_{I}(i,j) = (1-3\mu)\, w_{M}(i,j) \\ r_{P}(i,j), \text{if } w_{I}(i,j) = (1-2\mu)\, w_{P}(i,j) \\ r_{Q}(i,j), \text{ otherwise} \end{cases}, \forall i,j \geq 0 \tag{3.27}$$

$$r_{P''}(i,j) = \begin{cases} r_I(i-1,j), \text{if } w_{P''}(i,j) = \beta\, w_I(i-1,j) \\ r_{P''}(i-1,j), \text{otherwise} \end{cases} , \forall i \geq 1\, \forall j \geq 0 \tag{3.28}$$

$$r_E(i,j) = \begin{cases} r_I(i,j), \text{ if } w_E(i,j) = (1-\beta)\, w_I(i,j) \\ r_{P''}(i,j), \text{ otherwise} \end{cases} , \forall i \geq 0\, \forall j \geq 0 \tag{3.29}$$

$$t_{P''}(i,j) = \begin{cases} i-1, \text{ if } w_{P''}(i,j) = \beta\, w_I(i-1,j) \\ t_{P''}(i,j), \text{ otherwise} \end{cases} , \forall i \geq 1, \forall j \geq 0 \tag{3.30}$$

$$t_E(i,j) = \begin{cases} i, \text{ if } w_E(i,j) = (1-\beta)\, w_I(i,j) \\ t_{P''}(i,j), \text{ otherwise} \end{cases} , \forall i \geq 0, \forall j \geq 0 \tag{3.31}$$

At the end of these recursions, the subcurve giving the closest match to Q according to the Viterbi algorithm is given by $P|_{[a:b]}$ where

$$a = r_E(m,n)$$
$$b = t_E(m,n)$$

3.3 Importance of the PHMM method

Even in writer-specific scenarios, random variations in the writer's handwriting lead to variability in shape among different instances of the same word. This is the reason why we only look for approximate curve matching in handwriting search. Curve matching approaches based on Fréchet distance or DTW distance use a user-specified threshold ϵ and consider two (sub)curves as matching if the distance between the two (sub)curves is less than ϵ. However, it is often the case there is no value of ϵ that works well for all queries. Experiments show that there are situations where none of the top 5 closest matches returned by the DTW or Fréchet based algorithms are correct matches; in other words, there can be cases where the distance

measure does not reflect the shape similarity in the sense that is expected for search in handwriting. This is an inherent limitation of non-probabilistic approaches. They are generally restricted in the extent and nature of random variations in handwriting under which the can detect matches correctly. On the other hand, a pair HMM such as the one that we propose models explicitly the random variations in the shape of words. The parameters of the PHMM, namely the transition and emission probabilities, can be tuned using a training algorithm so that the probabilistic model it represents is fairly similar to the actual model underlying the variations in the writer's handwriting. The PHMM-based search technique is therefore likely to perform better in terms retrieval accuracy and precision vs recall rates.

Another advantage of our PHMM approach is that the parameters can be tuned to perform more specialized types of searches. For instance, by setting the parameter α to 0 in the PHMM shown in figure 3.1, we can constrain the algorithm to perform prefix-based search, i.e. look only for those words whose prefix matches with the query string. Similarly, by setting β to 0 in the PHMM, we can make the algorithm perform a suffix-based search, i.e. look for suffixes of words that match the query. The value of parameters η and α can be tuned to make the algorithm look for matches that occur closer to the beginning of a word rather than in the middle or near the suffix.

3.4 The language-neutral search technique

We now explain our language-neutral search technique. Given a handwritten document and a handwritten query string *s* both written by the same user, the algorithm must return the occurrences of that string in the given document. The partial matching distance between a sample curve and the query curve can be computed using one of the following distance measures ($D_{partial}$) - $\mathbf{PFD}_{\mathbf{x},\mathbf{y}}$, $\mathbf{PFD}_{\mathbf{y},\theta}$, $\mathbf{PDTW}_{\mathbf{x},\mathbf{y}}$, $\mathbf{PDTW}_{\mathbf{y},\theta}$ or using the Viterbi probabilistic score obtained on the PHMM as a similarity score. In case a distance measure is used, we use the negative of the distance as being a similarity score.

Since we are only looking for an approximate similarity of shapes, either a threshold T_{dist} or a number N is specified. We consider the threshold approach only for distance measures as

it is more intuitive for a lay user to specify a bound on the distance between curves rather than on a probabilistic score unless the user has a good understanding of the PHMM employed. The search algorithm reads the handwritten document and segments it using a custom word-segmentation algorithm to obtain a list of words $w_1, w_2, ..., w_n$. The query string as well as the words in the list are represented as finite-length sequences (arrays) of (x, y) values and hence define the corresponding polygonal curves. They are all normalized so that they all have the same height. Next if the distance measure (or similarity measure) used is $\mathbf{PFD}_{\mathbf{y},\theta}$, $\mathbf{PDTW}_{\mathbf{y},\theta}$ or the PHMM Viterbi score, we compute the (y, θ) values corresponding to each sample point of a curve and use the polygonal curve formed by these (y, θ) instead of the original curve in the rest of the search process. In case N was specified, the algorithm does the following:

- For each w_i in the list, find the similarity score ($score[i]$) as follows:
  If the PHMM method is used, then set
  $score[i] = ViterbiScore_{PHMM}(w_i, s)$
  else (a distance measure is used) compute the partial distance
  $dist[i] = D_{partial}(w_i, s)$, and set
  $score[i] = -dist[i]$
- Sort w_i's in the decreasing order of their corresponding scores, the $score[i]$'s
- Return the top N words from the sorted list as being N closest matches to s in that order

Else if T_{dist} was specified, then the algorithm creates and empty *output* list and does the following until there are no more words in the list to be checked for match:

- Let w_i denote the next word in the list that is yet to be checked for match
- Compute $dist[i] = D_{partial}(w_i, s)$
- If $dist[i] < T_{dist}$ then add w_i to the *output* list

The *output* list is the list of closely matching words found by the search algorithm.

Chapter 4

Applications and Experimental Results

4.1 Experiments

Unipen train_r01_v07 dataset: The first set of experiments used data from category #8 of the Unipen [5] train_r01_v07 dataset, a well-known standard dataset that contains handwritten data collected using various types of handwritten input devices from a vast number of people with widely differing writing styles. Among all benchmarks in train_r01_v07 dataset, only benchmark #8 is a realistic, application-oriented test since the word segmentation problem too should be solved by the recognition or search technique[1]. The data from the 'bbd' group was chosen as it has the most number of sentences - 858 English sentences written by 7 writers, containing a total of almost 16450 words. Also the sentences are realistic as most of them are excerpts from news reports (see figure 4.1). *We are not aware of any previous related work that was evaluated against category 8 of Unipen dataset*. Our work possibly presents the *first results on this realistic benchmark*, not only for our own approach but also for the most widely used DTW.

Multilingual dataset: Our second dataset consisted of sentences in 3 Asian languages (Hindi, Kannada, and Tamil) as well as a few icons/symbols written by two users. There were over 300 words, over 170 of them being words in Kannada/Hindi words or symbols/icons and the

[1] Source: http://ontolingua.nici.kun.nl/scrawls/unipen-pub-guide.html

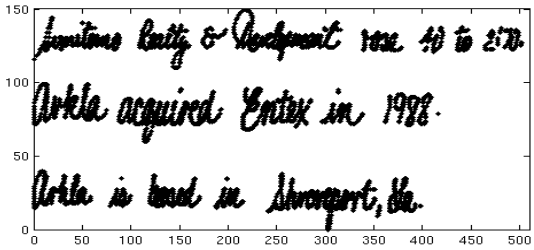

Figure 4.1: Some sentences written by a user in the Unipen dataset

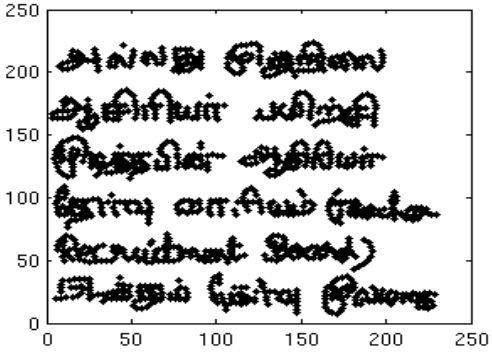

Figure 4.2: A sentence (containing Tamil and English words) from the multilingual dataset

remaining being Tamil words with Arabic numerals interspersed. A sample of the data is plotted in figure 4.2.

4.1.1 Preprocessing

All handwritten documents in the UNIPEN benchmark#8 dataset are a collection of sentences rather than words. Since our queries are substrings of words or whole words, sentences need to be split into individual words before our partial-matching algorithms can be applied. To segment sentences, we employed an algorithm that identifies indices of time series where the *split distance* d_s is greater than some threshold τ. The *split distance* at index i of the time series is defined as,

$$d_s = \min_{j>i}(x_j) - \max_{j<=i}(x_j) \tag{4.1}$$

where x_j is the x-coordinate corresponding to index j of the time series, i.e. at time instant t_j. An example of the output of our segmentation algorithm is shown in figure 4.3.

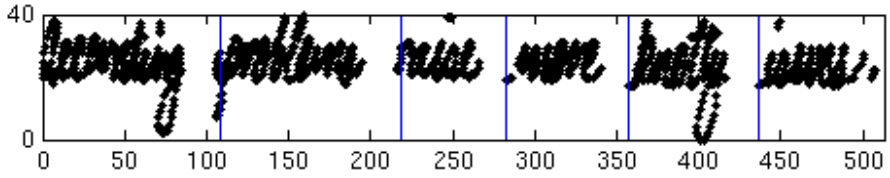

Figure 4.3: Splitting algorithm applied on a sentence, with τ=10. Vertical lines indicate the x-axis values of the first point in each word

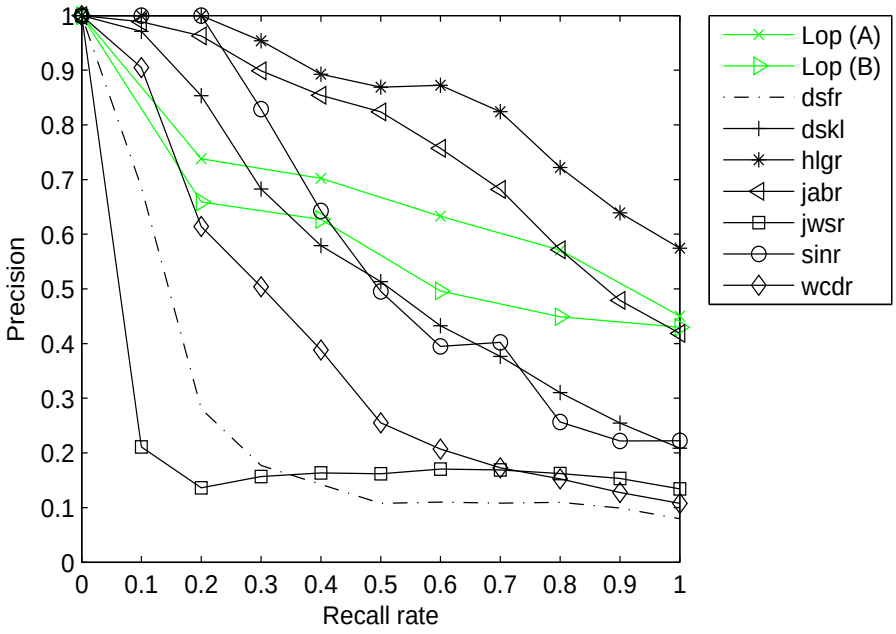

Figure 4.4: Average precision-recall on UNIPEN dataset for $\mathbf{PFD}_{\mathbf{x,y}}$

4.2 Results

In this project we concentrate on writer-dependent search tasks. Thus for each writer the query strings were obtained from the document written by the same writer. Each query string was created by selecting from the (x,y,t)-sequence of a sentence a subsequence corresponding to a substring of a word in that sentence. Words having less than 3 characters and words occurring less than 4 times in the dataset were not considered for forming queries. On an average around 20 query strings were selected for testing per writer. We performed similar experiments for studying the efficacy of the five distance measures - $\mathbf{PFD}_{\mathbf{x,y}}$, $\mathbf{PFD}_{\mathbf{y},\theta}$, $\mathbf{PDTW}_{\mathbf{x,y}}$,

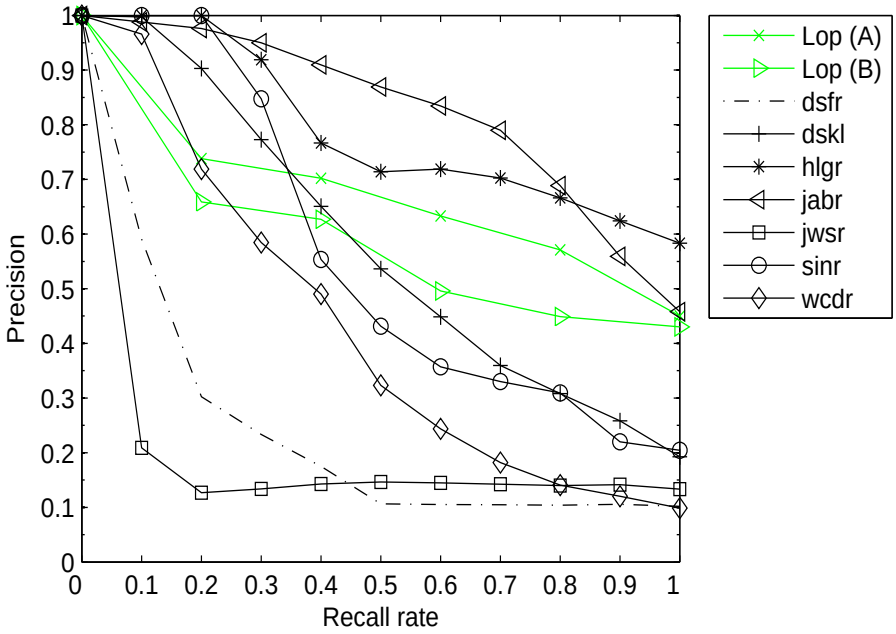

Figure 4.5: Average precision-recall on UNIPEN dataset for $\mathbf{PDTW}_{\mathbf{x},\mathbf{y}}$

$\mathbf{PDTW}_{\mathbf{y},\theta}$ and the PHMM score. For each sentence in the dataset, we found the distance (one of the four distance measures) between each word in it and the query word and assigned the minimum of these as the "distance" of the sentence. The sentences were then ranked in the increasing order of their distances and the precision and recall values calculated at each position (rank). Figures 4.4, 4.5, 4.6, 4.7 and 4.8 show the graph of average precision-vs-recall values for each UNIPEN user for different distance measures - $\mathbf{PFD}_{\mathbf{x},\mathbf{y}}$, $\mathbf{PDTW}_{\mathbf{x},\mathbf{y}}$, $\mathbf{PFD}_{\mathbf{y},\theta}$, $\mathbf{PDTW}_{\mathbf{y},\theta}$ and Viterbi algorithm on PHMM respectively. The graphs also show the precision-recall values reported by [7] for two writers, since it is a known related work that reports precision-recall for search on handwritten text, though their setting is simpler as they do not focus of partial search ability and also their results are on a custom dataset, not UNIPEN. It is seen that our PHMM based algorithm outperforms other distance measures.

For the multilingual dataset, we found the word-level retrieval accuracy for searches based on four distance measures - $\mathbf{PFD}_{\mathbf{x},\mathbf{y}}$, $\mathbf{PFD}_{\mathbf{y},\theta}$, $\mathbf{PDTW}_{\mathbf{x},\mathbf{y}}$, $\mathbf{PDTW}_{\mathbf{y},\theta}$. We used all instances of all words in the dataset as queries and noted the closest match retrieved by the search algorithm from the *rest of the dataset, leaving out the instance* being queried. Since the dataset

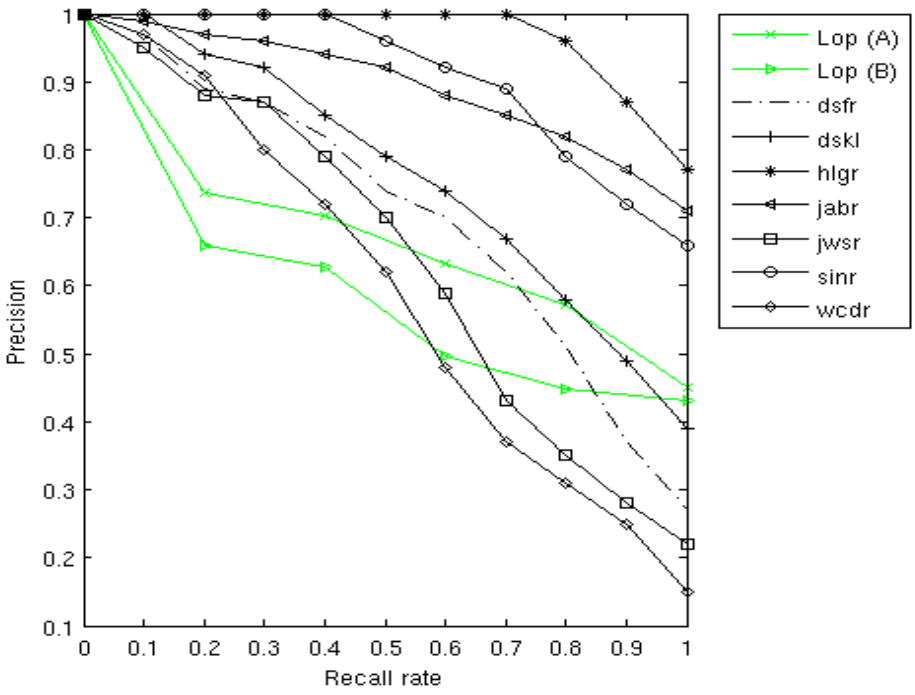

Figure 4.6: Average precision-recall on UNIPEN dataset for $\mathbf{PFD}_{\mathbf{y},\theta}$

was created such that it had exactly two instances of any word, we could define retrieval error rate as the probability of that the closest match found is not the other instance of that word, i.e. a wrong match. Thus average error is $e = M/N$ where M is the total number of mistakes and N is the number of words in the dataset. Table 4.2 shows the average retrieval accuracy on the data written by two users, for different distance/similarity measures.

4.3 A search application for Paper documents

We used our search algorithms to develop a search application for Paper documents on Simputer. The Paper documents contains the handwritten data as sequence of strokes. Our application converts the strokes into time series before searching. The application provides an interface for handwriting the query. The query *sketch* is also read as a time series (or just a sequence of x-y values, as needed). One of the four different distance measures can be selected. The application tries to match the query with the words obtained from the time series and ranks

Recall rate	PHMM	$\mathbf{PFD}_{\mathbf{y},\theta}$	$\mathbf{PDTW}_{\mathbf{y},\theta}$	$\mathbf{PFD}_{\mathbf{x},\mathbf{y}}$	$\mathbf{PDTW}_{\mathbf{x},\mathbf{y}}$
0.0	1.000	1.000	1.000	1.000	1.000
0.1	0.996	0.983	0.999	0.823	0.822
0.2	0.983	0.942	0.988	0.692	0.718
0.3	0.968	0.917	0.975	0.600	0.634
0.4	0.952	0.874	0.957	0.523	0.527
0.5	0.930	0.819	0.929	0.461	0.447
0.6	0.913	0.758	0.908	0.421	0.408
0.7	0.894	0.688	0.888	0.391	0.373
0.8	0.870	0.617	0.855	0.326	0.337
0.9	0.836	0.535	0.800	0.282	0.290
1.0	0.748	0.451	0.716	0.249	0.253

Table 4.1: Average precision at different recall rates on the UNIPEN dataset for various distance/similarity measures

Distance measure	Average accuracy (writer A)	Average accuracy (writer B)
PHMM	77.326%	99.167%
$\mathbf{PDTW}_{\mathbf{y},\theta}$	73.256%	97.500%
$\mathbf{PDTW}_{\mathbf{x},\mathbf{y}}$	47.674%	59.167%
$\mathbf{PFD}_{\mathbf{y},\theta}$	65.697%	81.667%
$\mathbf{PFD}_{\mathbf{x},\mathbf{y}}$	44.186%	64.167%

Table 4.2: Average word-retrieval accuracies on the multilingual dataset using different approaches

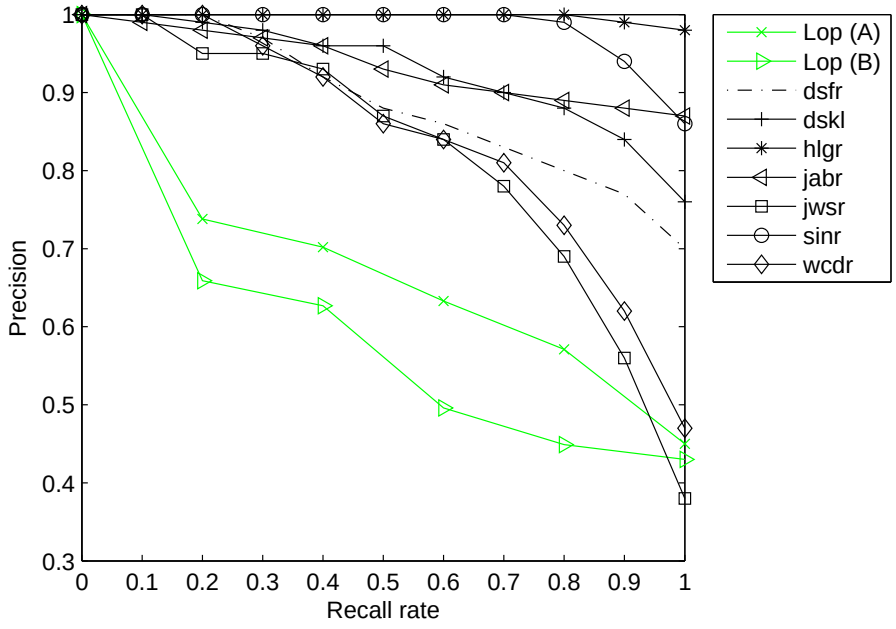

Figure 4.7: Average precision-recall on UNIPEN dataset for $\mathbf{PDTW}_{\mathbf{y},\theta}$

them. The application provides a sketchbox where the closely matching word from each file is displayed on the press of a button. The interfaces were developed using the Picowidget library provided by Picopeta.

The snapshots of the search application as well as sample Paper documents are shown. The application gave promising results on multilingual documents as well as on documents with free-form sketches and doodles. For efficiency purposes, our application uses a bounding box to restrict the search to only words/figures that are within a certain size. Figures 5.1, 5.2 and 5.3 show snapshots of Paper documents and our search application.

4.4 A searchable multilingual PhoneBook application

We also developed an application on Simputer for creating and searching a multilingual handwritten phonebook. The interfaces were developed using the X11 library. The application provides an interface for adding entries by handwriting. Since the entries are stored in *digital ink* format, they can be in any language. Another interface allows the user to enter by

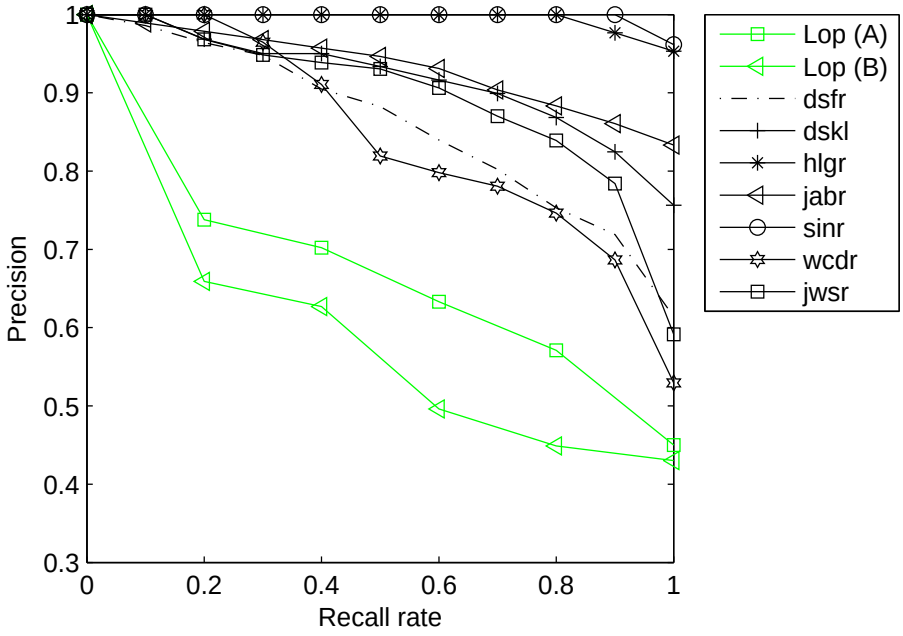

Figure 4.8: Average precision-recall on UNIPEN dataset for PHMM based on (y, θ) feature

handwriting a part or whole of the name to be searched. The distance measure to be used can be specified by selecting the corresponding radiobutton. The search application computes the distance between the query and each of the names entered in the phonebook and displays the top 3 closest matching entries.

Chapter 5

Discussion and Future Work

In this project we proposed DP-based approaches for searching online handwritten documents to find words matching a query string handwritten by the user. Our methods provide the flexibility of searching for words with a user-specified subword and can be implemented on PDAs. We implemented a search application based on our algorithms for searching Paper documents on Simputer and got encouraging results. We also proposed a pair HMM as a probabilistic model of partial curve matching. Our experimental results show that the PHMM-based search algorithm outperforms those based on the proposed variants of the partial Fréchet distance and the partial DTW - namely $\mathbf{PFD}_{\mathbf{x},\mathbf{y}}$, $\mathbf{PFD}_{\mathbf{y},\theta}$, $\mathbf{PDTW}_{\mathbf{x},\mathbf{y}}$ and $\mathbf{PDTW}_{\mathbf{y},\theta}$.

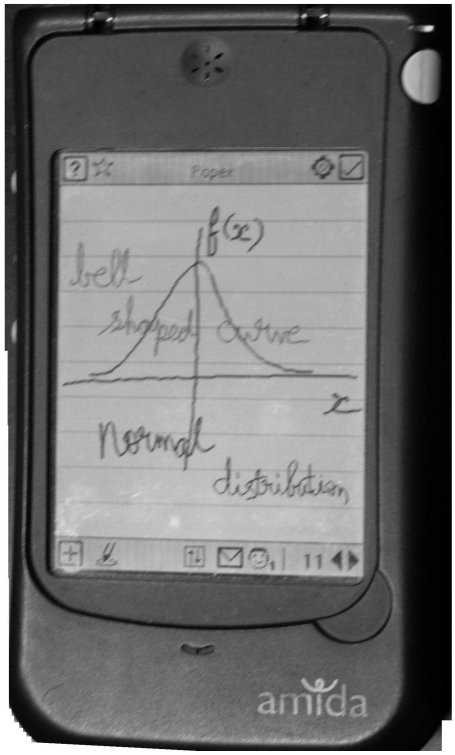

Figure 5.1: A sample Paper document with free-form writing

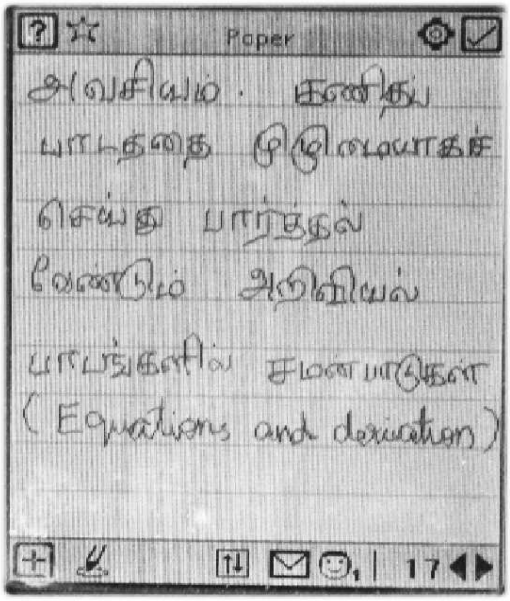

Figure 5.2: A sample Paper document with multilingual handwritten text

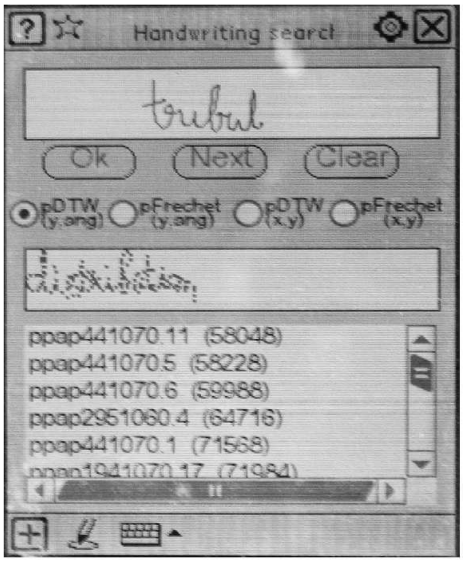

Figure 5.3: Search application for Paper documents on Simputer

Bibliography

[1] Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *Int. J. Comput. Geometry Appl*, 5:75–91, 1995.

[2] Boris Aronov, Sariel Har-Peled, Christian Knauer, Yusu Wang, and Carola Wenk. Fréchet distance for curves, revisited. In *Algorithms - ESA 2006, 14th Annual European Symposium, September 11-13, 2006, Proceedings*, volume 4168 of *Lecture Notes in Computer Science*, pages 52–63. Springer, 2006.

[3] Richard Durbin, Sean R. Eddy, Anders Krogh, and Graeme J. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.

[4] A. Efrat, Q. Fan, and S. Venkatasubramanian. Curve matching, time warping, and light fields: New algorithms for computing similarity between curves. *Journal of Mathematical Imaging and Vision*, 27(3):203–216, April 2007.

[5] I. Guyon, L. Schomaker, R. Plamondon, M. Liberman, and S. Janet. UNIPEN project of on-line data exchange and recognizer benchmarks. In *International Conference on Pattern Recognition*, pages B:29–33, 1994.

[6] Anil K. Jain and Anoop M. Namboodiri. Indexing and retrieval of on-line handwritten documents. In *ICDAR*, pages 655–659. IEEE Computer Society, 2003.

[7] Daniel Lopresti and Andrew Tomkins. On the searchability of electronic ink. In *Fourth International Workshop on Frontiers in Handwriting Recognition, Taipei, Taiwan, December 1994, Proceedings*, pages 156–165, 1994.

[8] Axel Mosig and Michael Clausen. Approximately matching polygonal curves with respect to the Fréchet distance. *Comput. Geom,* 30(2):113–127, 2005.

[9] Günter Rote. Computing the Fréchet distance between piecewise smooth curves. *Comput. Geom. Theory Appl.*, 37(3):162–174, 2007.

[10] H. Sakoe and S. Chiba. A dynamic programming approach to continuous speech recognition. In *Proc. Int. Cong. Acoustics*, Budapest, Hungary, Paper 20C-13, 1971.

[11] R. C. Veltkamp. Shape matching: similarity measures and algorithms. In Bob Werner, editor, *Proceedings of the International Conference on Shape Modeling and Applications (SMI-01)*, pages 188–199, Los Alamitos, CA, May 7–11 2001. IEEE Computer Society.

[12] Chris Watkins. Dynamic alignment kernels. Technical Report CSD-TR-98-11, Department of Computer Science, University of London, Egham, England, January 1999.