Exploring the Use of Conditional Random Field Models and HMMs for Historical Handwritten Document Recognition

Shaolei Feng, R. Manmatha and Andrew McCallum *
Center for Intelligent Information Retrieval
University of Massachusetts
Amherst, MA, 01003
[slfeng, manmatha, mccallum]@cs.umass.edu

Abstract

In this paper we explore different approaches for improving the performance of dependency models on discrete features for handwriting recognition. Hidden Markov Models have often been used for handwriting recognition. Conditional random fields (CRF's) allow for more general dependencies and we investigate their use. We believe that this is the first attempt at apply CRF's for handwriting recognition. We show that on the whole word recognition task, the CRF performs better than a HMM on a publicly available standard dataset of 20 pages of George Washington's manuscripts. The scale space for the whole word recognition task is large - almost 1200 states. To make CRF computation tractable we use beam search to make inference more efficient using three different approaches. Better improvement can be obtained using the HMM by directly smoothing the discrete features using the collection frequencies. This shows the importance of smoothing and also indicates the difficulty of training CRF's when large state spaces are involved.

1 Introduction

Although handwritten document recognition is a classical vision problem and has been researched for a long time, it is far from being solved. Good results have been achieved for online handwriting recognition, which takes full advantages of the dynamic information in strokes obtained using special input devices like tables. However, dynamic information is unavailable for huge volumes of valuable handwritten documents, for example, George Washington's letters at the Library of Congress, Issac Newton's papers at Cambridge University and the collection of Joseph Grinnell in the Museum of Vertebrate Zoology at U.C.Berkeley. Efficiently accessing and reading them requires advanced off-line handwriting recognition techniques. Off-line handwriting recognition is a harder problem. While good success has been achieved for small-vocabulary and highly constrained domains such as mail sorting and check processing, large vocabulary recognition on modern and historical data is still a challenging task [3, 10, 5]. A lot of work in handwriting recognition has been done at the character-level, where it is necessary to determine character boundaries - since character boundaries are difficult to determine this is sometimes done by jointly segmenting and recognizing the characters. In this project, all the data for training and testing are from a corpus of significantly degraded historical documents – manuscripts of George Washington. As in many other handwritten documents, the poor quality makes character boundaries hard to determine. So we directly recognize the entire word without character segmentation, as [5] did, and the recognition problem is formulated as a problem of labelling observation sequences on a large-vocabulary of words.

In this paper, we mainly focus on the discrete features since we want to compare the properties of Conditional Random Fields (CRF's) and HMMs on this same task and directly using continuous features for CRFs is still problematic. We first investigate using HMMs with discrete features for holistic word recognition in historical handwritten documents, where given a sequence of discrete feature sets a HMM estimates the most possible word sequence that could have generated it [1]. We assume that both transition probabilities and generative probabilities are subject to multinomial distributions. However, our experiments shows a HMM with discrete features without any smoothing for these features doesn't perform that well. To improve the recognition results, one way is to use a similar but more powerful model, such as a CRF model.

Recent research on machine learning shows CRFs have advantages over HMMs on tasks which involve labelling se-

*This work was supported in part by the Center for Intelligent Information Retrieval, by the National Science Foundation under grant NSF IIS-9909073 and by SPAWARSYSCEN-SD under grant N66001-02-1-8903.

[1] CRF's require a feature to be monotonically distributed. This is not true of the continuous features from which we derive our discrete features. Further it is not clear how one would smooth continuous features. For both these reasons we restrict our study to discrete features

quence data. On this kind of task, generative models such as HMM define a joint probability over observation and label sequences, which theoretically requires enumeration of all possible observation sequences. CRFs, as conditional undirected graphic models, model the conditional probabilities of label sequences given an observation sequence. In other words they don't involve generating a testing observation sequence. Furthermore, CRFs allow arbitrary dependencies on the observation sequence and have comparable computational efficiency if dependencies within state sequences are constrained. So in this paper, we investigate using CRFs for this task and compare them to HMMs and maximum entropy. We believe this is the first use of CRFs in this domain.

CRFs have been successfully used for many other domains, e.g. part of speech tagging [6], biomedical named entity recognition [8] and table extraction [9]. But all these applications are over a small state space. One challenge faced by CRFs on our task is the large amount of computation when training and decoding over the large state space - there is a state corresponding to each word (almost 1200 states). Linear-chain CRFs have similar computation complexity (although actual running times are much larger) as HMMs through analogous model optimization and decoding algorithms: CRFs use a similar forward-backward sum-product pass during model optimization to compute expectations and employ an analoge of Viterbi to decode the test observation sequence. To accommodate this model to the domain with a large state space, we use beam search to purge the state set when optimizing parameters and decoding observation sequences. Beam search is basically an optimization of best first search algorithm where only a fraction of paths are kept at each level. HMMs with Viterbi beam search [14] have been widely used in speech recognition to deal with large state space problem. Beam search methods can have a fixed width of beam or adaptive width based on different criterations, e.g. the cost(or probability) of each state and the K-L divergence [16]. Here, we test and compare three kinds of beam search methods based on N-best, ratio threshold and K-L divergence respectively.

We show on a publicly available database of 20 pages of George Washington's papers that beam search can drastically cut down computational time without hurting performance too much. The CRF model performs better than the HMM model with the same discrete features and a maximum entropy model.

Besides a CRF model, another way to improve the original HMMs with discrete features is to use an appropriate smoothing technique to estimate the probabilities of features generated by each word. It has been shown for a number of different tasks that smoothing can affect performance in a substantial manner [7]. The maximum likelihood estimate(MLE) for a HMM is often biased when the sample is relatively small. Smoothing using background probabilities of each discrete feature plays a role in the bias-variance tradeoff. In this way, our experiments show that the HMM model may be made to give better results than the CRF. This is because while CRFs are much more powerful than HMMs, many more parameters need to be estimated. In theory this can probably be solved by using a lot more training data. Practical limitations on obtaining large amounts of training data and computational constraints limit the use of large state space CRFs.

The remainder of this paper organized as follows: the next section give a brief overview of related work on handwriting recognition and the work on pruning state space in graphic models. Then we describe our HMM with discrete features for this task. In the following sections, we investigate two methods to improve our original HMM. We first introduce CRFs, a similar but more advanced model, and its specialization to our task. This is followed by a discussion of three different three different kinds of beam search models that can reduce the computation required for a large state CRF. We finally discuss smoothing techniques to improve the performance of our original HMM. Next, we present our experiments and conclude the report with a discussion of the results and future work.

1.1 Related Work

Although online handwriting recognition has advanced to the level of commercial application, offline handwriting recognition has only been successful in small-vocabulary and highly constrained domains [17, 13]. Only very recently have people started to look at offline recognition of large vocabulary handwritten documents [3]. Marti et al [10] proposed to use a Hidden Markov model (HMM) for modern handwritten material recognition. Each character is represented using a Hidden Markov model with 14 states. Words and lines are modelled as a concatenation of these Markov models. A statistical language model was used to compute word bigrams and this improved the performance by 10%. Vinciarelli et al [3, 4]used a sliding window HMM with an -n-gram model (up to trigrams) and demonstrated good results on modern large vocabulary handwritten data. Rath et al [5] focused on recognizing historical handwritten manuscripts using HMMs with one state for each word. By adding word bigrams from similar historical corpora they showed that the performance could approach an accuracy of 60%. Feng and Manmatha [11] take word recognition as a multi-class classification problem and survey the performance of a bunch of classification models on it, including naive Bayes models with kernel density estimate, maximum entropy models and support vector machines. Their experiments show that with Gaussian kernel density estimate naive Bayes models outperform some more sophisticated models such as support vector machines. This is not surprising because the Gaussian kernel density emphasizes the local information provided by each instance, which has been shown to be very useful in multimedia data analysis. Boosted decision trees [15] have also been shown to produce really good recognition results on the same dataset. Edwards et al [12] show that for Latin manuscripts one can apply gHMMs for recognizing words. Unfortunately, their Latin manuscripts seem to show much less variation than the test data here.

Results from information retrieval [1] show that for print optical character recognition (OCR), the retrieval performance doesn't drop significantly even for high word error rate. By analogy although the output will not satisfy the standard for human reading, we believe it is useful for handwriting retrieval based on text queries. Rath et al [19] also demonstrate a technique based on relevance models for historical manuscript retrieval which do not require any recognition. Finally word spotting techniques [20] have also been demonstrated for searching handwritten data.

2 Word Recognition with HMMs

We first test a HMM based on discrete features. As a generative model, a HMM estimates the joint probability of a hidden state sequence and a given observation sequence, which in our task are a sequence of words $S =< s_1, s_2, \ldots, s_T >$ and a sequence of discrete feature vectors $O =< o_1, o_2, \ldots, o_T >$ extracted from word images respectively:

$$P(S, O) = \prod_{t=0}^{T} P(s_t|s_{t-1})P(o_t|s_t) \qquad (1)$$

where T denotes the length of the sequences, and both transition probabilities $P(s_t|s_{t-1})$ and generative probabilities $P(o_t|s_t)$ are assumed to be subject to multinomial distributions. For each discrete feature in the feature vector $o_t =< o_{t1}, o_{t2}, \ldots, o_{tm} >$ extracted from the t-th word image, we assume it is independent of others given a hidden word s_t. Thus we have $P(o_t|s_t) = \prod_{i=0}^{m} P(o_{ti}|s_t)$. Given labelled handwritten documents as training set τ, these probabilities can be easily computed using maximum likelihood estimation (MLE), which gives very simple forms to calculate them. Let w and v be two arbitrary words from vocabulary V, the transition probabilities are calculated as:

$$P(s_t = w|s_{t-1} = v) = \frac{\sharp(\text{word pair}(v, w) \text{ occurs in } \tau)}{\sharp(\text{word } v \text{ occurs in } \tau)}$$

Let I_w denotes all the word images labelled as w in the training set τ, the generative probabilities are calculated as

$$P(o_{ti}|s_t = w) = \frac{\sharp(\ o_{ti} \text{ occurs as a feature of } I_w\ )}{\sharp(\text{all features of } I_w\ )} \qquad (2)$$

The estimation of transition probabilities is done as in [5] and includes with an averaging over the background distributions of these labels to smooth the probabilities:

$$\hat{P}(s_t|s_{t-1}) = \frac{1}{2}P(s_t|s_{t-1}) + \frac{1}{2}P(s_t) \qquad (3)$$

where $P(s_t)$ is the background probability of label s_t in the collection τ and calculated as:

$$\hat{P}(s_t = w) = \frac{1}{2} \cdot \frac{\sharp(w \text{ in } \tau\ )}{\sharp(\text{all words in } \tau)} + \frac{1}{2} \cdot \frac{1}{|V|} \qquad (4)$$

where $|V|$ is the size of the whole vocabulary.

Experiments in section 6.3 show this model doesn't perform that well. One way to improve the performance is to use a similar but more advanced model. In the next section we investigate such a model – CRF – for this task.

3 Conditional Random Fields Framework

A CRF [6] is defined as an undirected graphical model used to calculate the probability of a possible label sequence conditioned on the observation sequence. The structure of random fields basically could be an arbitrary graph obeying the Markov property. Let $O =< o_1, o_2, \ldots, o_T >$ and $S =< s_1, s_2, \ldots, s_T >$ denote the observation sequence and the label sequence respectively (In general CRFs, T need not be the same for O and S). A CRF formulates the conditional probability of S given O as:

$$P_\theta(S|O) = \frac{1}{Z_\theta(O)} \prod_q \left(\exp\left(\sum_k \lambda_k f_k(\mathbf{s}_q, \mathbf{o}_q) \right) \right) \qquad (5)$$

where feature functions $\{f_k\}$ are defined on any subset of the random variables in the sequences $\mathbf{s}_q \subset S$, $\mathbf{o}_q \subset O$, λ_k is learned weight for each feature function, and Z is a normalization factor over all possible state sequences:

$$Z_\theta(O) = \sum_{S \in S^T} \prod_q \left(\exp\left(\sum_k \lambda_k f_k(\mathbf{s}_q, \mathbf{o}_q) \right) \right) \qquad (6)$$

In the simplest case, the graph is an undirected linear chain among output states, where CRFs make a first-order Markov independence assumption. Under this configuration, equation (5) is rewritten as:

$$P_\theta(S|O) = \frac{1}{Z_\theta(O)} \exp\left(\sum_{t=1}^{T} \sum_k \lambda_k f_k(s_t, s_{t-1}, O, t) \right) \qquad (7)$$

A feature function (as distinct from an image feature) is defined over the the current state, the previous state and image features computed over the whole observation sequence. Usually feature functions are binary predicate. For example, assume that the only image feature used is the length of the current word image. Then, the feature function f_k is 1 if the current state corresponds to "Washington", the previous state to "Colonel" and the length of the current word is 8, else $k_k = 0$. Note that even in the simple case the number of weights is $\boldsymbol{O}(|\mathsf{S}|^2|\mathsf{O}|)$.

To reduce the number of parameters estimated, we simplify the model into a conditionally-trained hidden Markov model, in which all the incoming transitions into a state will share the same weight and only at each separate step of the sequence we create weights for the current state and observation. In this case, the conditional probability becomes:

$$P_\theta(S|O) = \frac{1}{Z_\theta(O)} \exp\left(\sum_{t=1}^{T}\left(\sum_k(\lambda_k f_k(s_t, O, t)) + \sum_l(\mu_l g_l(s_t, s_{t-1}))\right)\right) \quad (8)$$

The number of parameters becomes $\boldsymbol{O}(|\mathsf{S}||\mathsf{O}| + |\mathsf{S}|^2)$.

3.1 Inference and Training in CRFs

Inference in CRFs is done as follows: Given an observation sequence $\tilde{O}$, from all possible label(state) sequences find the one $\tilde{S}$ with the largest conditional probability over the distribution of $P(S|\tilde{O})$. This distribution is defined by the undirected graphic structure and the set of weights. Note the number of possible state sequences is exponential in the sequence length T. For an arbitrarily-structured CRF, it is intractable to calculate the normalization factor in equation (6). In HMM-Style CRFs, the normalization factor becomes:

$$Z_\theta(O) = \sum_{S \in S^T} \exp\left(\sum_{t=1}^{T}\left(\sum_k(\lambda_k f_k(s_t, O, t)) + \sum_l(\mu_l g_l(s_t, s_{t-1}))\right)\right) \quad (9)$$

A dynamic programming algorithm like Viterbi decoding can be used to efficiently calculate the normalization factor.

The parameters $\theta = \{\lambda \ldots\}$ are estimated through optimizing the model over a training set consisting of labelled sequences, $\boldsymbol{D} = \{O^{(i)}, S^{(i)}\}_{i=1}^{N}$, i.e. try to find the set of weights maximizing the log-likelihood of the labelled sequences in the training set:

$$L = \sum_{i=1}^{N} \log(P(S^{(i)}|O^{(i)})) - \sum_k \frac{\lambda_k^2}{2\sigma^2} \quad (10)$$

where the second term is a Gaussian prior over parameters smoothing over the training data [18].

Iterative scaling [6] is a general method to optimize parameters of CRFs and other exponential models. Sha and Pereira [21] use the limited memory quasi-Newton (L-BFGS) [22] method instead, which is shown to be several orders of magnitude faster than iterative scaling. Like iterative scaling, L-BFGS is also a gradient based optimization procedure but only requires the first-derivative of the objective function. The gradient of the likelihood function is [9]:

$$\begin{aligned} \frac{\delta L}{\delta \lambda_k} &= \sum_{i=1}^{N} C_k(S^{(i)}, O^i) - \\ & \sum_{i=1}^{N}\left(Z(O^{(i)}) \sum_t \alpha_t \Phi_t \beta_t^T\right) - \frac{\lambda_k}{\sigma^2} \end{aligned}$$

where $C_k(S, O) = \sum_t f_k(s_t, s_{t-1}, O, t)$ is the count of feature k given O and S, $\Phi_t = C(s_t, s_{t-1}, O, t)$, and α_t and β_t are the forward and backward vectors transferred throughout the linear chain. That means the optimization procedure requires forward-backward inference at each iteration for each training sequence. Note the computations of forward-backward algorithm are $\boldsymbol{O}(|\mathsf{S}|^2 T)$, so when the state space is large the optimization procedure will be extremely slow.

Here, every word is taken as a state so there could be thousands of states. In the next section we discuss applying beam search for CRFs to significantly speed up the forward-backward procedure.

3.2 Training and Inference with Beam Search

The basic idea of beam search is simple. At each stage of the trellis for inference before passing any message to the next stage we first purge the states at this stage and keep only a small fraction of them. The number of states kept is usually called the width of beam. So when using beam search for forward-backward procedure, the number of outgoing transitions from the current stage to the next stage will dramatically drop. Our goal is to prune as many as possible states while minimize the loss of performance. To determine the states to eliminate, we need some criterion. Based on different criteria for purging states, the beam search method works differently. Note that we talk about the probabilities of states here, but actual implementations of inference and forward-backward algorithm use costs, which are equal to the negative logarithm of the probabilities. In the implementation of beam search, these costs need to be converted into probabilities.

1. **N-best Beam Search**

   The simplest way to do beam search is to sort all the states in the current stage according to their probabilities in descending order. Then only the top K states are kept and the other states are eliminated.

2. **Ratio Threshold based Beam Search**

   At stage i, we first determine the maximal probability P_i^m of all the states $P_i^m = \max_s P_i(s)$. Then a dynamic threshold is calculated based on the value P_i^m:

   $$\tau_i = \frac{P_i^m}{K} \tag{11}$$

   where K is a empirically selected constant. Then all states s' at this stage whose $P_i(s') < \tau_i$ will be eliminated.

   This method doesn't have a fixed width of the beam at each stage and the criterion for purge is based on the individual probability of every state. This method is widely used with HMMs in speech recognition [23].

3. **K-L Divergence based Beam Search**

   Pal et al [16] recently present a novel beam search method based on K-L Divergence. The basic idea is to approximate single variable potentials with a constrained adaptively sized sum of Kronecker delta functions and minimize the KL divergence between the approximated distribution and its original. At each stage of the trellis for Viterbi inference or forward-backward procedure, the probabilities of all the states form some arbitrary discrete probability distribution, say p. Any subset of these states, indexed with $I = \{1, \dots, k\}$, forms some other distribution which could be approximated as a sum of weighted and normalized Kronecker deltas, say q. The goal is to find the subset of these states which minimize the K-L divergence between p and q. Pal et al[16] show this K-L divergence is equal to the negative logarithm of the sum of the probabilities of the subset states. More formally, suppose we want to find the minimal subset of states such that the K-L divergence $KL(q||p) \leq \epsilon$, then that implies minimizing $|I|$ s.t.

   $$KL(q||p) = -\log \sum_{i \in I} p_i \leq \epsilon \tag{12}$$

   Now it can be solved by sorting the states according to their probabilities in a descending order and then selecting the states from the top until the sum of their probabilities satisfies equation (12).

4 Word Recognition with CRFs

Using CRFs for word recognition is quiet straightforward when the data are given as labelled handwritten documents. Handwritten documents are segmented into word images. Each word image is an observation and its corresponding label is the value of its state in CRFs.

The ideal case in each instance is a labelled sentence. However this is intractable for degraded handwritten documents because important clues for sentences, such as punctuations are faded or connected with words resulting in a failure to detect them. In our case each sequence instance is a completely labelled page, with a length within 200 $\sim$ 300 words. The drawback of using pages as training instances is that unreliable transitions between connections of two separate sentences will be involved and learned by the model.

Because both the size of the state space and the length of sequences in our project are large, we use the HMM-Style CRFs described by equation (8) in section 3.

Continuous image features are first extracted from each word image based on its scalar and shape. Each continuous feature is quantized into a fixed number of bins. The set of discretized features of each word image is its observation representation. Details on image features are given in section 6.1.

The model features are defined in a straightforward way. For example $f_k(s_t, O, t)$ is equal to 1 if the word image at position t is labelled as "Fredericksburgh" and its length is at level 10 (the highest level for our discretized image features), otherwise it is zero. The transition features $g_k(s_t, s_{t-1})$ are defined in a similar manner. For example $g_k(s_t, s_{t-1})$ is equal to 1 if the word image at position t is labelled as "Fredericksburgh" and the previous word is "defend", otherwise zero.

5 Feature Probability Smoothing for HMMs

Besides the CRF model, we explore using smoothing techniques to improve the performance of our original HMM model in section 2 - note that we are smoothing the features here not the words as is usually done. The maximum likelihood estimate for generative probabilities in equation 2 is prone to be bias when the sample size is relative small. To alleviate this bias we smooth the generative probabilities using background probabilities of discrete features. In stead of a direct averaging as in [5], we tune the weight to optimize the likelihood on a held-out portion of a training sample. The formulation for feature probability smoothing has a linear form as follows:

$$\hat{P}(o_{ti}|s_t) = (1 - \lambda)P(o_{ti}|s_t) + \lambda P(o_{ti}) \tag{13}$$

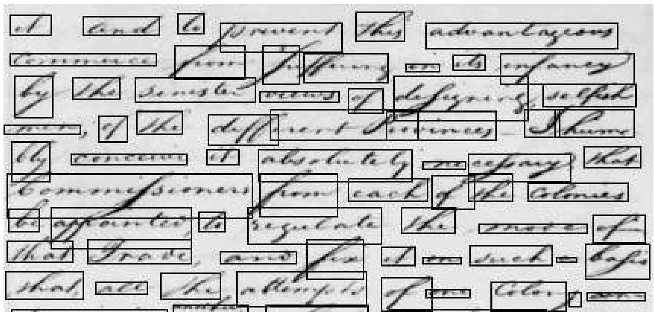

Figure 1: A part of one segmented page in our dataset.

where $P(o_{ti})$ is the background probability of discrete feature o_{ti} in the training set τ, directly calculated as the frequency of o_{ti} in τ. λ is the parameter of this linear smoothing and tuned through optimizing the likelihood on a validation set created from a portion of the training sample.

6 Experimental Results

6.1 Experimental Setup

Our evaluation dataset consists of 20 pages from a collection of letters by George Washington. This is a publicly available standard dataset (see Lavrenko et al [5]). Each page is accurately segmented into individual word images, each of which has been manually transcribed. We don't lowercase transcribed words, so "*region*" and "*Region*" are taken as two different words. There are 4865 words in the corpus in total and 1187 of them are unique. Figure 1 shows a part of a segmented page in our dataset.

We used the feature set in [5]. 27 continuous features are extracted from each word image, which consists of 6 scalar features(i.e. height, width, aspect ratio, area, number of descenders in the word, and number of ascenders in the word) and 21 profile-based features which are obtained through a discrete Fourier Transform(DFT) over three time series generated from the word image, which are projection profile, upper word profile and lower word profile. To discretize the continuous features, two overlapping binning schemes are used. The first divides each feature dimension into 10 bins while the second creates additional 9 bins shifted by half a bin size. So totally we have 52 discrete features for each word image. Please refer to [5, 19] for the feature details.

We use word accuracy rate as our performance measure, i.e. the proportion of the words that are recovered exactly as they were in the manual transcript. 20-fold cross-validation is used to get a stable performance evaluation. Each iteration leaves one page for test, and trains the model over the other 19 pages. We use the mean accuracy rate as the final evaluation measure. Since our dataset is relative small, many words in the test set don't occur in any training pages - these are called out-of-vocabulary(OOV) terms as in [5] and cause errors of the recognition. We use two types of mean accuracy rate – mean accuracy rate with OOVs and mean accuracy rate without OOVs.

Since our data are from a collection of natural language documents(letters), the frequency of words can be approximated by a Zipf distribution, in which a few words have very high frequencies, and most words occur very infrequently. Over our whole dataset, 681 words have only one occurrence; 1008 words have less than 5 occurrences each but 30 words have 1856 occurrences in total. The unbalance and sparsity of training data for different words make the problem difficult.

6.2 Tune and Compare Beam Search for Our CRF Model

All the three kinds of Beam Search in section 3.2 require us to experimentally decide the parameters controlling the width of beam. For this purpose, we select two pages from our handwritten documents and use them as a training and a test example respectively. Even in this small dataset with 486 words in total, there are 264 states(unique words). On average there are less than 2 instances for each state, which means our model has been very starved for training data. But for this project, this is the smallest unit we can use for tuning, including more pages results in a sharp increase of the run time.

Table 1, Table 2 and Table 3 show the results using different values of tuning parameters for N-best, ratio threshold and K-L divergence based beam search respectively. As the tables show the accuracy changes non-linearly with the tunint parameters. In certain regions it is realtively insensitive while in others it is very sensitive. The tables only show some values of the parameters - mostly those very large changes occur.

Since with comparable accuracy N-best runs much slower than the other two methods, we did not do experiments using N-best over the whole dataset. We select $K = 1.01$ and $\epsilon = 0.75$ as the parameters of ratio threshold and KL-divergence respectively when testing over the whole dataset.

6.3 Results on the Whole Dataset

The final test of our CRF model on the whole dataset using CRFs with the ratio threshold takes roughly one day on a Intel Xeon 2.80GHz machine for one test instance while with K-L divergence it takes 18 hours. Our HMMs using discrete features are tested on the same feature set.

Table 4 show the results on the whole dataset using CRFs with ratio threhold based beam search and K-L divergence

Fixed Beam Width	10	80	105	106	107	132	264
Accuracy w/o OOV	0.001	0.001	0.001	0.645	0.645	0.645	0.645
Run Time (in Secs)	60	131	140	142	142	153	2944

Table 1: N-best Beam Search with different fixed beam widths

K in Equation (11)	1.0001	1.001	1.01	1.1	1.2	1.5	2
Accuracy w/o OOV	0.505	0.518	0.645	0.645	0.645	0.645	0.645
Run Time (in Secs)	97	99	107	1127	1238	1340	1527

Table 2: Ratio Threshold Beam Search with different K values

ϵ in $KL \leq \epsilon$	0.9	0.8	0.79	0.77	0.75	0.5	0
Accuracy w/o OOV	0.001	0.001	0.475	0.584	0.645	0.645	0.645
Run Time (in Secs)	62	70	75	87	91	209	2944

Table 3: KL Divergence Beam Search with different ϵ in $KL \leq \epsilon$

Accuracy Rate	with OOV	w/o OOV
Maximum Entropy with discrete features (in [11])	0.416	0.494
HMM with discrete features	0.336	0.404
HMM with discrete features after smoothing	0.504	0.595
CRFs with Ratio threshold beam search	0.417	0.503
CRFs with K-L divergence beam search	0.428	0.525
HMM with continuous features (in [5])	0.497	0.586

Table 4: Results and comparison on the whole dataset. CRFs and Maximum Entropy cannot really be used with the continuous features described here and so are not directly comparable with HMMs using continuous features.

based beam search respectively and a comparison with a Maximum Entropy model and HMM models. All discrete features for these models are the same. For the Maximum Entropy model, the model features are defined as those in CRFs for observational-state pairs, only observation and state at the same position are considered (see [11] for details). From the results, CRFs with a K-L divergence based beam search outperforms that with a ratio threshold based beam search by a small margin. Both CRFs outperform the Maximum Entropy model, showing the importance of transition information - CRFs have them while the maximum entropy model does not use them. The HMM with discrete features where the features are not smoothed does not perform that well (the words are smoothed for all HMMs). HMM performance can be improved substantially by also smoothing the features and as can be seen this makes them better than the CRF's. For reference, CRFs and Maximum Entropy use some kind of Gaussian prior for smoothing [18]. However, we believe that the poorer performance of CRFs is due to the substantially larger number of parameters that need to be estimated. In addition all the parameters are estimated at the same time while the probabilities for HMM's are estimated separately in this special case. More training data might improve the results but there are significant difficulties in using more training data. First, creating large amounts of training data is labor intensive and expensive. Second, CRFs are much slower and hence this would also require large amounts of computation. An alternative approach to increasing the amount of training data required would be to drastically reduce the state space. This would probably require dropping the whole word paradigm and moving to a character based approach with its attendant segmentation difficulties. We have so far compared all techniques on the same features. Continuous features can substantially improve performance. As a point of reference we show the performance of an HMM using continous features modeled by a single Gaussian [5]. As can be seen, this performs better than all the other techniques. Even better performance may be obtained using continuous features (see for example [11]). However, directly using continuous features for CRFs is still problematic. CRFs require the continuous features to have a monotonic distribution. Most of the continuous features used in the paper here are not monotonic and in general it is non-trivial to find such features. Using the existing non-monotonic continuous features with CRFs leads to poor performance.

7 Conclusions and Future Work

This paper is the first application of CRFs to word recognition in handwritten documents, and one where a large state space is involved. Because even for linear-chain CRFs the computation is quadratic with the number of states, standard CRFs cannot finish the training and inference procedure in reasonable time. To make CRFs practical on this task, we investigate different beam search methods and their performance. We successfully integrated it with CRFs to solve large state space problems and dramatically speeded up the decoding and optimization procedure with minimum effect on performance. We show that a CRF performs better than a HMM on the same discrete feature set if the HMM is not smoothed. However, by smoothing with background collection probabilities we can substantially improve the performance of the HMM over the CRF showing that a more sophisticated model is not the only criterion for good performance. We suspect that in this case the CRF performs more poorly because it has many more parameters and hence may require much more training data. One approach to solving this problem is to reduce the state space and move to character based CRFs and we will investigate this in the future. Other possible extensions include initializing transition weights through external data from similar text corpus, which will improve the estimation of transition parameters and also speed up the model optimization procedure. More sophisticated features might also help.

References

[1] S. M. Harding, W. B. Croft and C. Weir Probabilistic Retrieval of OCR Degraded Text Using N-Grams, In *Proc. of the 1st European Conference on Research and Advanced Technology for Digital Libraries*. 1997, pp. 345-359.

[2] Eugen C. Buehler, Lyle H.Ungar Maximum Entropy Methods for Biological Sequence Modeling, In *Workshop on Data Mining in Bioinformatics of KDD01*, 2001.

[3] A. Vinciarelli, S. Bengio and H. Bunke Offline Recognition of Large Vocabulary Cursive Handwritten Text. in *The Proc. of ICDAR'03*, pp. 1101-1105.

[4] A. Vinciarelli, S. Bengio and H. Bunke Offline Recognition of Unconstrained Handwritten Texts Using HMMs and Statistical Language Models. In *IEEE Transactions PAMI*, vol. 26, no. 6, 2004, pp. 709–720.

[5] Lavrenko, V., Rath, T. and Manmatha, R., Holistic Word Recognition for Handwritten Historical Documents in *the Proc. of DIAL'04*, pp. 278-287.

[6] J. Lafferty,A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML, 2001.*

[7] C. Zhai and J. Lafferty. A Study of Smoothing Methods for Language Models Applied to Information Retrieval. *ACM Trans. on Information Systems*, 22(2):179-214,2004.

[8] B. Settles Biomedical Named Entity Recognition Using Conditional Random Fields and Rich Feature Sets. In *Proc. of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications(NLPBA)*, 2004.

[9] D. Pinto, A. McCallum, X. Wei, W.B. Croft Table Extraction Using Conditional Random Fields In *the Proc. of ACM SIGIR'03*

[10] U.-V. Marti and H. Bunke Using a Statistical Language Model to Improve the Performance of an HMM-Based Cursive Handwriting Recognition System. in *the Jnl. of Pattern Recognition and Artificial Intelligence*, I 15:1 (2001) 65-90.

[11] S.L. Feng and R. Manmatha, Classification Models for Historical Documents Recognition. *In the Proc. of ICDAR'05*, pp. 528–532.

[12] J. Edwards , Y. Whye Teh, D. Forsyth, R. Bock, M. Maire and G. Vesom Making Latin Manuscripts Searchable using gHMMs, In *the Proc. of NIPS 2004*

[13] S. Madhavnath and V. Govindaraju The Role of Holistic Paradigms in Handwritten Word Recognition *IEEE Trans. on PAMI*, vol 23, no. 2, 2001, pp. 149–164.

[14] Mosur K. Ravishankar. Efficient Algorithms for Speech Recognition. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1996.

[15] N. Howe, T. Rath, and R. Manmatha Boosted Decision Trees for Word Recognition in Handwritten Document Retrieval. In *the Proc. of ACM SIGIR'05*, pp. 377–383.

[16] C. Pal, C. Sutton, A. McCallum Constrained Kronecker Deltas for Fast Approximate Inference and Estimation. *submitted to UAI 2005.*

[17] Plamondon and S. N. Srihari On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey In *IEEE Trans. on PAMI* vol 22, no. 1, 2000, pp. 63–84.

[18] S. F. Chen and R. Rosenfeld. A Gaussian prior for smoothing maximum entropy models. Technical Report CMU-CS-99-108, CMU, 1999.

[19] T. Rath, R. Manmatha and V. Lavrenko. A Search Engine for Historical Manuscript Images in *the Proceedings of SIGIR'04*, pp. 297 -304

[20] T. Rath and R. Manmatha Word Image Matching Using Dynamic Time Warping. In *the Proceedings of CVPR'03*, vol. 2, 2003, pp. 521–527

[21] F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proceedings of Human Language Technology, NAACL, 2003.*

[22] J.Norcedal and S.J. Wright. *Numerical Optimization*. Springer, 1999.

[23] F. Jelinek Statistical Methods for Speech Recognition *MIT Press*