AN ABSTRACT OF THE THESIS OF

Guohua Hao for the degree of Master of Science in Computer Science presented on February 27, 2009.

Title: Revisiting Output Coding for Sequential Supervised Learning

Abstract approved: ______________________________

Alan Fern

Markov models are commonly used for joint inference of label sequences. Unfortunately, inference scales quadratically in the number of labels, which is problematic for training methods where inference is repeatedly preformed and is the primary computational bottleneck for large label sets. Recent work has used output coding to address this issue by converting a problem with many labels to a set of problems with binary labels. Models were independently trained for each binary problem, at a much reduced computational cost, and then combined for joint inference over the original labels. Here we revisit this idea and show through experiments on synthetic and benchmark data sets that the approach can perform poorly when it is critical to explicitly capture the Markovian transition structure of the large-label problem. We then describe a simple cascade-training approach and show that it can improve performance on such problems with negligible computational overhead.

Revisiting Output Coding for Sequential Supervised Learning

by

Guohua Hao

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented February 27, 2009
Commencement June 2009

Master of Science thesis of Guohua Hao presented on February 27, 2009.

APPROVED:

Major Professor, representing Computer Science

Director of the School of Electrical Engineering and Computer Science

---

Dean of the Graduate School

Guohua Hao, Author

ACKNOWLEDGEMENTS

First of all, I would like to thank my major advisor, Dr. Alan Fern, for his tremendous support and patience in helping me complete this project. I would also like to thank Dr. Thomas Dietterich for his helpful discussions on this thesis work, and Dr. John Langford for discussion of the counter example to independent SECOC. Finally, I would like to thank National Science Foundation for supporting this project by grant IIS-0307592.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF TABLES

Chapter 1 – Introduction

In this chapter, we first formulate the sequential supervised learning problem and describe its application in a variety of domains. Then we describe the research direction on which this thesis focuses within the sequential supervised learning framework and give a brief review of previous work that has been done in this direction. Finally, the contributions of this thesis are summarized, and the outline of this thesis is given at the end.

1.1 Sequential Supervised Learning

Many applications of machine learning involve assigning a class label to each element of an input sequence collectively. For example, in natural language processing, the task of part-of-speech (POS) tagging is to label each word in a sentence with a part of speech tag ("noun", "verb" etc.) (Ratnaparkhi, 1996). In computational biology, the task of protein secondary structure prediction is to assign a secondary structure class to each amino acid residue in the protein sequence (Qian and Sejnowski, 1988).

These kinds of problems can be formulated as follows:

Given: A set of training examples of the form (X_i, Y_i), where each $X_i = (\mathbf{x}_{i,1}, \ldots, \mathbf{x}_{i,T_i})$ is a sequence of T_i feature vectors and each $Y_i = (y_{i,1}, \ldots, y_{i,T_i})$ is a corresponding sequence of class labels, where $y_{i,t} \in \mathcal{Y} = \{1, \ldots, L\}$.

Find: A classifier H that, given a new sequence X of feature vectors, predicts the corresponding sequence of class labels $Y = H(X)$ accurately.

These problems are called Sequential Supervised Learning (SSL) or Label Sequence Learning (LSL) problems (Dietterich, 2002).

One of the most famous SSL problems is the NETtalk task, in which English words are pronounced by assigning a phoneme and stress to each letter of the word (Sejnowski and Rosenberg, 1987). Other applications of SSL arise in information extraction (McCallum et al., 2000), handwritten word recognition (Taskar et al., 2004), and so on.

We study sequential supervised learning problems where the number of labels is large. Markov models are widely used for sequential supervised learning. However, for a first-order Markov model, the time complexity of standard inference algorithms, such as Viterbi and forward-backward, scales quadratically with the number of labels, L. More generally, for an Nth order Markov model, the time cost scales as $O(T \cdot L^{N+1})$. When L number is large, this can be problematic when predictions must be made quickly. Even more problematic is the fact that a number of recent approaches for training Markov models, e. g. Lafferty et al. (2001), Tsochantaridis et al. (2004), and Collins (2002), repeatedly perform inference during learning. As the number of labels grows, such approaches quickly become computationally impractical.

1.2 Previous Work

This problem has led to recent work that considers various approaches to reducing computation while maintaining high accuracy. One approach has been to integrate approximate-inference techniques such as beam search into the learning process (Collins and Roark, 2004; Pal et al., 2006). A second approach has been to consider learning subclasses of Markov models that facilitate sub-quadratic inference—for example, using linear embedding models (Felzenszwalb et al., 2003), or bounding the number of distinct transition costs (Siddiqi and Moore, 2005). Both of these approaches have demonstrated good performance on a number of sequential supervised learning problems that satisfy the assumptions of the models.

A third recent approach, which is the focus of this thesis, is sequential error-correcting output coding (SECOC) (Cohn et al., 2005) motivated by the success of error-correcting output coding (ECOC) for non-sequential problems (Dietterich and Bakiri, 1995). The idea is to use an output code to "reduce" a multi-label sequence-learning problem to a set of binary-label problems. SECOC solves each binary problem independently and then combines the learned models to make predictions over the original large label set. The computational cost of learning and inference scales linearly in the number of binary models and is independent of the number of labels. Experiments on several data sets showed that SECOC significantly reduced training and labeling time with little loss in accuracy.

1.3 Contributions

While the initial SECOC results were encouraging, the study did not address SECOC's general applicability and its potential limitations. For non-sequential learning, ECOC has been shown to be a formal reduction from multi-label to binary label classification

(Langford and Beygelzimer, 2005). One contribution of this thesis is to show that this result does not hold for sequential supervised learning. That is, there are sequential supervised learning problems such that, for any output code, SECOC performs poorly compared to directly learning on the large label set, even assuming optimal learning for the binary problems.

Given the theoretical limitations of SECOC and the prior empirical success, the main goals of this thesis are the following: 1) to better understand the types of problems for which SECOC is suited, and 2) to suggest a simple approach to overcoming the limitations of SECOC and to evaluate its performance. We present experiments on synthetic and benchmark data sets. The results show that the originally introduced SECOC performs poorly for problems where the Markovian transition structure is important, but where the transition information is not captured well by the input features. These results suggest that, when it is critical to explicitly capture the Markovian transition structure, SECOC may not be a good choice.

In response, we introduce a simple extension to SECOC called cascaded SECOC where the predictions of previously learned binary models are used as features for later models. Cascading allows for richer transition structure to be encoded in the learned model with little computational overhead. Our results show that cascading can significantly improve on SECOC for problems where capturing the Markovian structure is critical.

1.4 Outline

The remainder of this thesis is organized as follows.

In chapter 2, we review the conditional random field model, which has been successfully applied to sequential supervised learning and which also serves as the base learning algorithm in this thesis. Two training algorithms of conditional random fields used in this thesis are outlined. Chapter 3 describes how to use the error-correcting output coding algorithm in standard supervised learning and how to generalize this idea in sequential supervised learning. Based on the analysis of the limitations of previous work, we propose the core algorithm of this thesis in chapter 4. Experimental results on both benchmark data sets and synthetic data sets are shown and analyzed in chapter 5. And chapter 6 concludes this thesis with summary and future work.

Chapter 2 – Conditional Random Fields

Conditional random fields (CRFs) proposed by Lafferty et al. (2001) have been widely applied to many sequential supervised learning problems with excellent results, such as POS tagging (Lafferty et al., 2001) and noun-phrase chunking (Sha and Pereira, 2003). In this chapter, we first give a short review of conditional random fields. Then two major algorithms for training CRFs are outlined, which will serve as the base training algorithms in this thesis.

2.1 A Short Review of CRFs

Let (X, Y) be a sequential labeled training example, where $X = (\mathbf{x}_1, \ldots, \mathbf{x}_T)$ is the observation sequence and $Y = (y_1, \ldots, y_T)$ is the sequence of labels, where $y_t \in \{1, \ldots, L)$ for all t. A conditional random field is a linear chain Markov random field (Geman and Geman, 1984) over the label sequence Y globally conditioned on the observation sequence X. For a first-order Markov mode, the Hammersley-Clifford theorem states that the conditional probability distribution $P(Y|X)$ has the following form:

$$P(Y|X) = \frac{1}{Z(X)} \exp\left[\sum_t \Psi_t(y_t, X) + \Psi_{t-1,t}(y_{t-1}, y_t, X)\right] \quad ,$$

where $\Psi_t(y_t, X)$ and $\Psi_{t-1,t}(y_{t-1}, y_t, X)$ are *potential functions* defined on cliques y_t and (y_{t-1}, y_t), which capture (respectively) the degree to which y_t is compatible with X and the degree to which y_t is compatible with a transition from y_{t-1} and with X. These potential functions can be arbitrary real-valued functions. The exponential function ensures that $P(Y|X)$ is positive, and the normalizing constant

$$Z(X) = \sum_{Y'} \exp\left[\sum_t \Psi_t(y'_t, X) + \Psi_{t-1,t}(y'_{t-1}, y'_t, X)\right]$$

ensures that $P(Y|X)$ sums to 1. Normally, it is assumed that the potential functions do not depend on t and can be represented as linear combinations of binary features (Lafferty et al., 2001):

$$\begin{aligned} \Psi_t(y_t, \mathbf{X}) &= \sum_\alpha \lambda_\alpha f_\alpha(y_t, \mathbf{X}) \\ \Psi_{t-1,t}(y_{t-1}, y_t, \mathbf{X}) &= \sum_\beta \gamma_\beta g_\beta(y_t, y_{t-1}, \mathbf{X}) \ , \end{aligned}$$

where parameters λ_α's and γ_β's are trainable weights, and functions f_α and g_β are boolean features. For example, in part-of-speech tagging $f_{123}(y_t, X)$ might be 1 when $\mathbf{x}_t$ is the word "bank" and y_t is the class "noun" (and 0 otherwise). It is natural to define features that depend only on a sliding window $w_t(X)$ centered at $\mathbf{x}_t$.

Many algorithms have been proposed for training CRFs. Below we describe two efficient training algorithms, which will be used in this thesis.

2.2 TREECRF

The TREECRF algorithm for training CRFs was first proposed by Dietterich et al. (2004). This algorithm uses functional gradient tree boosting to maximize the log-likehood of the training data.

2.2.1 Functional Gradient Tree Boosting

This algorithm was first proposed by Friedman (2001). Suppose we are solving a supervised learning problem which is defined as minimizing the following loss function:

$$\ell(\mathbf{x}, y, \Psi(\mathbf{x}, y)).$$

Instead of explicitly representing function Ψ by a set of parameters and computing gradient descent with respect to these parameters, Ψ is assumed to be represented as a sum of functions

$$\Psi_m = \Psi_0 + \Delta_1 + \cdots + \Delta_m$$

in functional space and Δ_m is the *functional gradient* computed as:

$$\Delta_m = -\eta_m \, E_{\mathbf{x},y} \left[\frac{\partial}{\partial \Psi(\mathbf{x}, y)} \ell(\mathbf{x}, y, \Psi(\mathbf{x}, y)) \right]_{\Psi(\mathbf{x},y)=\Psi_{m-1}(\mathbf{x},y)}.$$

It indicates how the function Ψ_{m-1} should change on all possible points $(\mathbf{x}, y)$ in order to decrease the loss function. In practice, the value of functional gradient descent is computed at each training example $(\mathbf{x}_i, y_i)$ sampled from some fixed but unknown distribution $P(\mathbf{x}, y)$, that is,

$$\Delta_m(\mathbf{x}_i, y_i) = \left[-\frac{\partial}{\partial \Psi(\mathbf{x}_i, y_i)} \sum_j \ell(\mathbf{x}_j, y_j, \Psi(\mathbf{x}_j, y_j)) \right]_{\Psi(\mathbf{x},y)=\Psi_{m-1}(\mathbf{x},y)}.$$

Based on the set of *functional gradient training examples* $((\mathbf{x}_i, y_i), \Delta_m(\mathbf{x}_i, y_i))$, it is possible to train a function $h_m(\mathbf{x}, y)$ to approximate the functional gradient $\Delta_m(\mathbf{x}_i, y_i)$. In particular, if $h_m(\mathbf{x}, y)$ is a regression tree that minimizes

$$\sum_i [h_m(\mathbf{x}_i, y_i) - \Delta_m(\mathbf{x}_i, y_i)]^2,$$

this method is called *functional gradient tree boosting*. Taking one step in the direction of this fitted function will give us

$$\Psi_m = \Psi_{m-1} + \eta_m h_m,$$

where η_m is the step size. Friedman (2001) suggests growing $h_m(\mathbf{x}, y)$ via a best-first version of the CART algorithm (Breiman et al., 1984) and stopping when the regression tree reaches a pre-set maximum number of leaves L. Overfitting is controlled by tuning L (e.g., by internal cross-validation).

2.2.2 TREECRF Algorithm

We can apply functional gradient tree boosting to train CRFs. Let

$$F^{y_t}(y_{t-1}, w_t(X)) = \Psi(y_t, w_t(X)) + \Psi(y_{t-1}, y_t, w_t(X)) \tag{2.1}$$

be a function that computes the "desirability" of label y_t given values for label y_{t-1} and input features $w_t(X)$. There are K such functions F^k, one for each class label $k \in \mathcal{Y}$. Then the CRFs have the form

$$P(Y|X) = \frac{1}{Z(X)} \exp \sum_t F^{y_t}(y_{t-1}, w_t(X)) \ .$$

Assume, without loss of generality, that there is only one occurrence of $w_t(X)$ in each sequence X. The functional gradient descent of the negative log-likelihood, which is chosen as the loss function, with respect to $F^k(k', w_d(X))$ can be computed as:

$$\begin{aligned} \Delta^k(k', w_d(X)) &= \frac{\partial \log P(Y|X)}{\partial F^k(k', w_d(X))} \\ &= I(y_{d-1} = k', y_d = k) - P(y_{d-1} = k', y_d = k \mid w_d(X)) \ , \end{aligned}$$

where $I(y_{d-1} = k', y_d = k)$ is the indicator function whose value is 1 if the transition $k' \rightarrow k$ is observed from position $d-1$ to position d in the sequence Y and 0 otherwise, and where $P(y_{d-1} = k', y_d = k \mid w_d(X))$ is the predicted probability of this transition according to the current potential functions.

This functional gradient can be interpreted as our error on a probability scale. If the transition $k' \rightarrow k$ is observed in the training example, then the predicted probability $P(y_{d-1} = k', y_d = k \mid w_d(\mathbf{X}))$ should be 1 in order to minimize the loss. If the transition is not observed, then the predicted probability should be 0. Based on the functional gradient training examples $((k', w_d(\mathbf{X})), \Delta^k(k', w_d(\mathbf{X})))$, a regression tree h^k can be generated to update function F^k. The pseudo code of the TREECRF algorithm is shown in Table 2.1.

The TREECRF algorithm is able to deal with feature interactions. Each regression tree can be viewed as defining several new feature combinations—one corresponding to each path in the tree from the root to a leaf. The resulting potential functions still have the form of a linear combination of features, but the features can be quite complex.

2.3 Voted Perceptron

The voted perceptron algorithm was first applied to train CRFs by Collins (2002). In this algorithm, Equation 2.1 is re-written as

$$F^{y_t}(y_{t-1}, w_t(X)) = \mathbf{w} \cdot \phi(\mathbf{x}_t, y_{t-1}, y_t) \; , \qquad (2.2)$$

where **w** is a weight vector and $\phi(\cdot)$ is a feature vector defined over clique (y_{t-1}, y_t) and conditioned on $\mathbf{x}_t$. Both of them are independent of position t. Let

$$\Phi(X, Y) = \sum_t \phi(\mathbf{x}_t, y_{t-1}, y_t)$$

Table 2.1: Pseudo code of TREECRF algorithm

```
TREEBOOST($Data, L$)
// $Data = \{(X_i, Y_i) : i = 1, \dots, N\}$
for each class $k$, initialize $F_0^k(\cdot, \cdot) = 0$
for $m = 1, \dots, M$
    for class $k$ from 1 to $K$
        $S(k)$ := GENERATEEXAMPLES($k, Data, Pot_{m-1}$)
            // where $Pot_{m-1} = \{F_{m-1}^u : u = 1, \dots K\}$)
        $h_m(k)$ := FITREGRESSIONTREE($S(k), L$)
        $F_m^k := F_{m-1}^k + h_m(k)$
    end
end
return $F_M^k$ for all $k$
end TREEBOOST

GENERATEEXAMPLES($k, Data, Pot_m$)
$S := \{\}$
for example $i$ from 1 to $N$
    execute the forward-backward algorithm on $(X_i, Y_i)$
        to get $\alpha(k,t)$ and $\beta(k,t)$ for all $k$ and $t$
    for $t$ from 1 to $T_i$
        for $k'$ from 1 to $K$
            $P(y_{i,t-1} = k', y_{i,t} = k \mid X_i) :=$
                $\frac{\alpha(k', t-1) \exp[F_m^k(k', w_t(X_i))] \beta(k,t)}{Z(X_i)}$
            $\Delta(k, k', i, t) := I(y_{i,t-1} = k', y_{i,t} = k) -$
                $P(y_{i,t-1} = k', y_{i,t} = k \mid X_i)$
            insert $((w_t(X_i), k'), \Delta(k, k', i, t))$ into $S$
        end
    end
end
return $S$
end GENERATEEXAMPLES
```

and

$$F(X,Y) = \sum_t F^{y_t}(y_{t-1}, w_t(X)) = \mathbf{w} \cdot \Phi(X,Y) \ \ ,$$

then the predicted label sequence $\hat{Y}$ for an observation sequence X is given as

$$\hat{Y} = \operatorname*{argmax}_{Y \in \mathcal{Y} \times \cdots \times \mathcal{Y}} F(X,Y).$$

The weight vector **w** is updated as follows

$$\mathbf{w} = \mathbf{w} + \Phi(X,Y) - \Phi(X,\hat{Y})$$

whenever $\hat{Y} \neq Y$.

2.4 Time Complexity

Most of the existing methods involve the repeated computation of the partition function $Z(X)$ and/or maximizing over label sequences, which is usually done using the forward-backward and Viterbi algorithms. The time complexity of these algorithms is $O(T \cdot L^{k+1})$, where L is the number of class labels, k is the order of Markov model, and T is the sequence length. So even for first order CRFs, training and inference scale quadratically in the number of class labels, which becomes computationally demanding for large label sets. In the next chapter, we describe the use of output coding for combatting this computational burden.

Chapter 3 – ECOC for Sequential Supervised Learning

In this chapter, we first describe the idea of using error-correcting output coding (ECOC) in standard supervised learning. Then we review one recent work which applies the ECOC idea to sequential supervised learning. At the end, we analyze the limitations of this recent work using a counter example.

3.1 Error-Correcting Output Coding (ECOC)

For non-sequential supervised classification, the idea of Error-Correcting Output Coding (ECOC) has been successfully applied to solve multi-class problems by Dietterich and Bakiri (1995). Suppose we wish to solve a multi-class classification problem where the training examples have the form $(\mathbf{x}_i, y_i)$, $i = 1, \ldots, N$ and $y_i \in \mathcal{Y} = \{1, \ldots, L\}$. The first step is to build a code matrix M for this problem as shown in Table 3.1. Each class label $j \in \mathcal{Y}$ is assigned a codeword C_j, which is a binary vector of length n. These code words are taken as rows in matrix M. The second step is to take each column b_k in M as a binary partition of the original label set $\mathcal{Y}$, that is, $b_k(y) \in \{0, 1\}$. Then a binary classifier h_k can be learned using training examples $\{(\mathbf{x}_i, b_k(y_i)) : i = 1 \ldots, N\}$. Hence, in ECOC algorithm, a multi-class learning problem is reduced to a number of binary-class learning problems.

To predict the label of a new instance $\mathbf{x}$, we concatenate the predictions of each

Table 3.1: ECOC code matrix and base classifiers

Class Label	Code Words				
	b_1	b_2	$\cdots$	$\cdots$	b_n
C_1	1	0	$\cdots$	$\cdots$	1
C_2	0	1	$\cdots$	$\cdots$	0
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
C_L	0	1	$\cdots$	$\cdots$	1
classifier	h_1	h_2	$\cdots$	$\cdots$	h_n

binary classifier to get a vector

$$H(\mathbf{x}) = (h_1(\mathbf{x}), h_2(\mathbf{x}), \ldots, h_n(\mathbf{x})).$$

The predicted label $\hat{y}$ is then given by

$$\hat{y} = \operatorname*{argmin}_{j \in \mathcal{Y}} \Delta(H(\mathbf{x}), C_j),$$

where Δ is some distance measure, such as the Hamming distance. In some implementations $H(\mathbf{x})$ stores probabilities rather than binary labels.

The issue of how to design the error-correcting code is discussed by Dietterich and Bakiri (1995).

Table 3.2: Pseudo code for training in SECOC

```
TRAINSECOC($Data, CodeMatrix, BinaryCRF$)
// $Data = \{(X_i, Y_i) : i = 1, \ldots, N\}$
// $CodeMatrix$ is of length $n$
// $BinaryCRF$ returns a binary-label CRF for a given training data set
Initialize the set $H$ to be empty set
for $k = 1, \ldots, n$
    Let $b_k$ be the $k$-th column in $CodeMatrix$
    Initialize $TempData$ to be empty set
    for $i = 1, \ldots, N$
        Insert $(X_i, b_k(Y_i))$ as a training example into $TempData$
    end
    // training a binary-label CRF $h_k$ based on data set $TempData$
    $h_k = BinaryCRF(TempData)$
    Insert $h_k$ into $H$
end
return $H$
end TRAINSECOC
```

3.2 Sequential Error-Correcting Output Coding (SECOC)

The ECOC idea has recently been applied to sequential supervised learning problems by Cohn et al. (2005) under the name sequential error-correcting output coding (SECOC), with the motivation of reducing computation time for large label sets. In SECOC, instead of training a multi-class CRF, we train a binary-class CRF h_k for each column b_k in the code matrix M. More specifically the training data for binary-class CRF h_k is given by $\{(X_i, b_k(Y_i))\}$, where

$$b_k(Y_i) = (b_k(y_{i,1}), b_k(y_{i,2}), \ldots, b_k(y_{i,T_i}))$$

is a binary label sequence. The pseudo code for SECOC training is given in Table 3.2.

Table 3.3: Pseudo code for testing in SECOC

```
TESTSECOC($X, CodeMatrix, H, \Delta$)
// $X = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)$ is an input observation sequence
// $CodeMatrix$ is of length $n$
// $H = (h_1, h_2, \ldots, h_n)$, where each $h_i$ is a binary-label CRF trained based on $CodeMatrix$
// $\Delta$ is some distance measure between two vectors
Let $ProbMatrix$ to be a $T$-by-$n$ matrix
for $k = 1, \ldots, n$
    let $prob = (p_1, p_2, \ldots, p_T)$
    // run forward-backward algorithm over $X$ using binary CRF $h_k$
    $p_t = P(y_t = 1|X, h_k), t = 1, \ldots, T$
    set the $k$-th column in $ProbMatrix$ as $prob'$
end
let $Y = (y_1, y_2, \ldots, y_T)$ be a 1-by-$T$ vector
for $t = 1, \ldots, T$
    let $prob$ be the $t$-th row in $ProbMatrix$
    $y_t = \text{argmin}_{j \in \mathcal{Y}} \Delta(prob, C_j)$ // $C_j$ is the $j$-th row in $CodeMatrix$
end
return $Y$
end TESTSECOC
```

Given a set of binary classifiers for a code matrix and an observation sequence X, we compute the multi-label output sequence as follows. First, we use the forward-backward algorithm on each h_k to compute the probability that each sequence position should be labeled 1. Then for each sequence element $\mathbf{x}_t$, we form a probability vector $H(\mathbf{x}_t) = (p_1, ..., p_n)$ where p_k is the probability computed by h_k for $\mathbf{x}_t$. After that, we let the label for $\mathbf{x}_t$ be the class label y_t whose codeword is closest to $H(\mathbf{x}_t)$ based on the L_1 distance. The pseudo code for this testing algorithm is given in Table 3.3. The complexity of this inference process in a first-order Markov model is just $O(n \cdot T)$ where n is the codeword length, which is typically much smaller than the number of labels squared. Thus, SECOC can significantly speed inference time for large label sets.

3.3 SECOC Counter Example

Prior work by Cohn et al. (2005) has demonstrated on several problems that SECOC can significantly speed training time without significantly hurting accuracy. However, this work did not address the potential limitations of the SECOC approach. A key characteristic of SECOC is that each binary CRF is trained completely independently of the others, and each binary CRF only sees a very coarse view of the multi-class label sequences. Intuitively, it appears difficult to represent complex multi-class transition models between y_{t-1} and y_t using such independent chains. This raises the fundamental question of the representational capacity of the SECOC model. The following counter example gives a partial answer to this question, showing that the SECOC model is unable to represent relatively simple multi-label transition models.

Consider a simple Markov model with three states $\mathcal{Y} = \{1, 2, 3\}$ and deterministic transitions $1 \rightarrow 2$, $2 \rightarrow 3$ and $3 \rightarrow 1$. A 3-by-3 diagonal code matrix

y	b_1	b_2	b_3
C_1	1	0	0
C_2	0	1	0
C_3	0	0	1

is sufficient for capturing all non-trivial codewords for this label set—that is, all non-trivial binary partitions of $\mathcal{Y}$. Below we show an example label sequence and the corresponding binary code sequences.

	y_1	y_2	y_3	y_4	y_5	y_6	y_7
Y	1	2	3	1	2	3	1
$b_1(Y)$	1	0	0	1	0	0	1
$b_2(Y)$	0	1	0	0	1	0	0
$b_3(Y)$	0	0	1	0	0	1	0

Given a label sequence $Y = \{1, 2, 3, 1, 2, 3, 1, \ldots\}$, a first-order Markov model learned for $b_1(Y)$ will converge to $P(y_t = 1|y_{t-1} = 0) = P(y_t = 0|y_{t-1} = 0) = 0.5$. It can be shown that as the sequence length grows such a model will make independent and identically distributed (iid) predictions according to the stationary distribution that predicts 1 with probability $1/3$. See Appendix A for details. The same is true for b_2 and b_3. Since the i.i.d. predictions between the binary CRFs are independent, using these models to make predictions via SECOC will yield a substantial error rate, even though the sequence is perfectly predictable. Independent first-order binary transition models are simply unable to capture the transition structure in this problem. Our experimental results will show that this deficiency is also exhibited in real data.

Chapter 4 – Cascaded SECOC Training of CRFs

Based on the analysis in Chapter 3, we know that each binary CRF in the SECOC algorithm has very limited knowledge about the Markovian transition structure. In this chapter, a simple extension of this algorithm is given to improve this situation.

4.1 c-SECOC algorithm

In order to better capture the Markovian transition structure, we can provide limited coupling between the binary CRFs in SECOC. One way to do this is to include observation features in each binary CRF that record the binary predictions of previous binary CRFs. We call this approach cascaded SECOC (c-SECOC), as opposed to the previous algorithm, which we will call independent SECOC (i-SECOC).

Assume the training examples are given as $S = \{(X_i, Y_i)\}$. Let $Y_i^{(j)}$ be the prediction of the binary CRF h_j learned with the j-th binary partition b_j in code matrix M, and $y_{it}^{(j)}$ be the t-th element of $Y_i^{(j)}$. To train the binary CRF h_k based on the k-th binary partition b_k, each training example (X_i, Y_i) is extended to $(X_i', b_k(Y_i))$, where each $\mathbf{x}_{it}'$ is the union of the observation features $\mathbf{x}_{it}$ and the predictions of the previous h binary CRFs at sequence positions from $t - l$ to $t + r$ ($l = r = 1$ in our experiments) except position t:

$$\mathbf{x}'_{it} = (\mathbf{x}_{it}, y^{(k-1)}_{i,t-l}, \ldots, y^{(k-1)}_{i,t-1}, y^{(k-1)}_{i,t+1}, \ldots, y^{(k-1)}_{i,t+r},$$
$$\ldots, y^{(k-h)}_{i,t-l}, \ldots, y^{(k-h)}_{i,t-1}, y^{(k-h)}_{i,t+1}, \ldots, y^{(k-h)}_{i,t+r}).$$

We refer to h as the cascade history length. We do not use the previous binary predictions at position t as part of $\mathbf{x}'_{it}$, since such features have significant autocorrelation which can easily lead to overfitting. To predict label sequence Y for a given observation sequence X, we make predictions from the first binary CRF to the last, feeding predictions into later binary CRFs as appropriate, and then use the same decoding process as i-SECOC.

Via the use of previous binary predictions, c-SECOC has the potential to capture Markovian transition structure that i-SECOC cannot. Our experiments in Chapter 5 show that this is important for problems where the transition structure is critical to sequence labeling but is not reflected in the observation features. The computational overhead of c-SECOC over i-SECOC is to increase the number of observation features, which typically has negligible impact on the overall training time. As the cascade history grows, however, there is the potential for c-SECOC to overfit with the additional features. We will discuss this issue further in the next chapter.

Chapter 5 – Experimental Results

We compare CRFs trained using i-SECOC, c-SECOC, and beam search over the full label set. We consider two existing base CRF training algorithms: *gradient-tree boosting (GTB)* (Dietterich et al., 2004) and *voted perceptron (VP)* (Collins, 2002). GTB is able to construct complex features from the primitive observations and labels, whereas VP is only able to combine the observations and labels linearly. Thus, GTB has more expressive power, but can require substantially more computational effort. In all cases we use the forward-backward algorithm to make label-sequence predictions and measure accuracy according to the fraction of correctly labeled sequence elements. We used random code matrices constrained so that no columns are identical or complementary, and no class labels have the same code word.

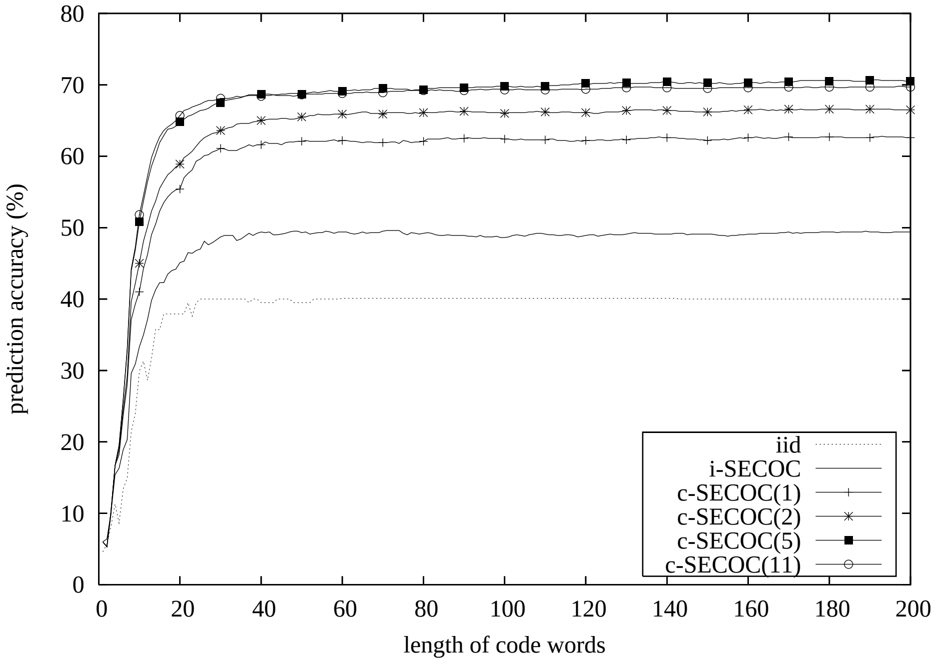
First, we consider a non-sequential baseline model denoted as “iid” which treats all sequence elements as independent examples, effectively using non-sequential ECOC at the sequence element level. In particular, we train iid using i-SECOC with zeroth-order binary CRF, that is CRFs with no transition model. This model allows us to assess the accuracy that can be attained by looking at only a window of observation features. Second, we denote by “i-SECOC” the use of i-SECOC to train first-order binary CRFs. Third, we denote by “c-SECOC(h)” the use of c-SECOC to train first-order binary CRFs using a cascade history of length h.

5.1 Summary of Results

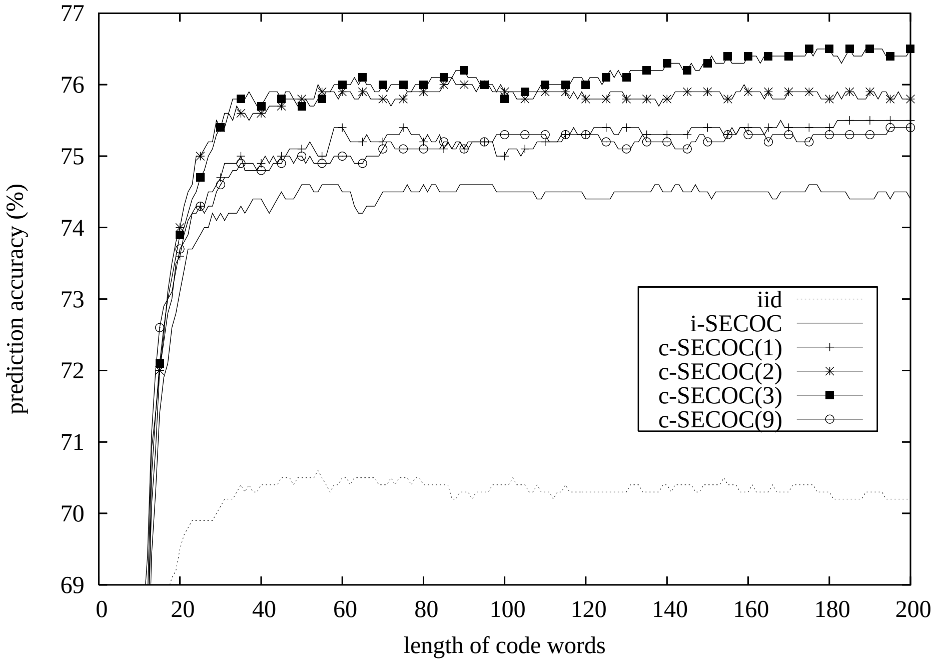
The results presented below justify five main claims: 1) i-SECOC can fail to capture significant transition structures, leading to poor accuracy. Such observations were not made in the original evaluation of i-SECOC (Cohn et al., 2005). 2) c-SECOC can significantly improve on i-SECOC through the use of cascade features. 3) The performance of c-SECOC can depend strongly on the base CRF algorithm. In particular, it appears critical that the algorithm be able to capture complex (non-linear) interactions in the observation and cascade features. 4) c-SECOC can improve on models trained using beam search when GTB is used as the base CRF algorithm. 5) When using weaker base learning methods such as VP, beam search can outperform c-SECOC.

5.2 Nettalk Data Set

The Nettalk task (Sejnowski and Rosenberg, 1987) is to assign a phoneme and stress symbol to each letter of a word so that the word is pronounced correctly. Here the observations correspond to letters yielding a total of 26 binary observation features at each sequence position. Labels correspond to phoneme-stress pairs yielding a total of 134 labels. We use the standard 1000 training and 1000 test sequences from Dietterich and Bakiri (1995).



(a) window size 1 with GTB



(b) window size 3 with GTB

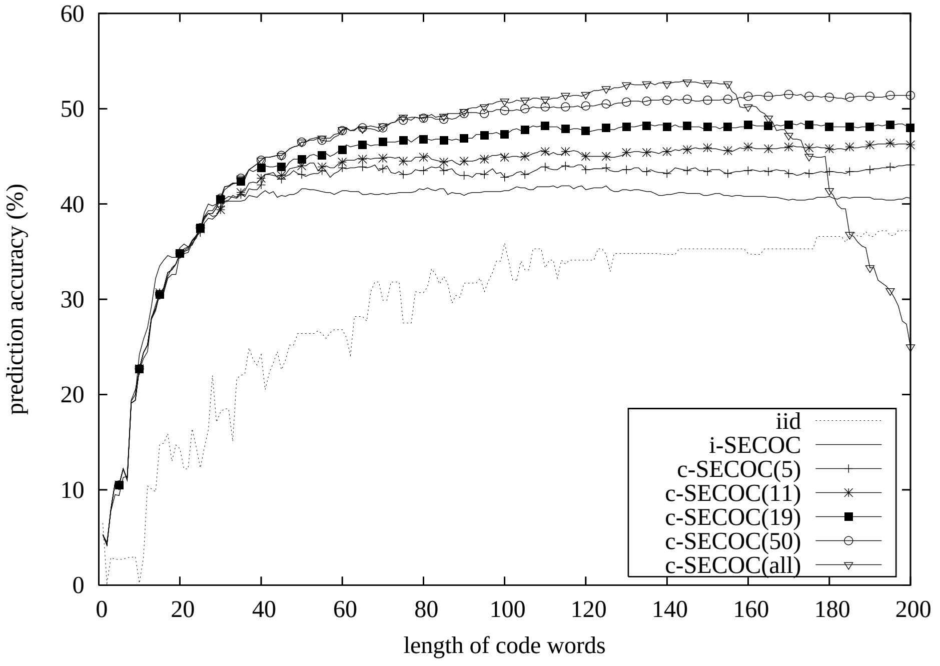
Figure 5.1: Nettalk data set with window sizes 1 and 3 trained by GTB.

5.2.1 Comparing to i-SECOC

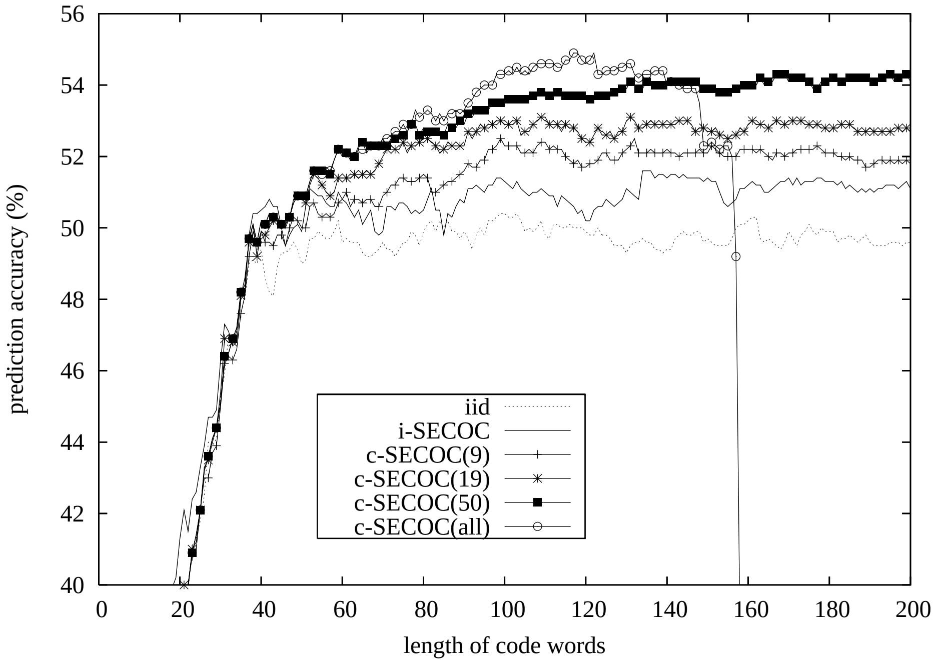
Figures 5.1a and 5.1b show the results for training our various models using GTB with window sizes 1 and 3. For window size 1, we see that i-SECOC is able to significantly improve over iid, which indicates that i-SECOC is able to capture useful transition structure to improve accuracy. However, we see that, by increasing the cascade history length, c-SECOC is able to substantially improve over i-SECOC. Even with $h = 1$, the accuracy improves by over 10 percentage points. This indicates that the independent CRF training strategy of i-SECOC is unable to capture important transition structure in this problem. c-SECOC is able to exploit the cascade history features in order to capture this transition information, which is particularly critical in this problem where apparently just using the information in the observation window of length 1 is not sufficient to make accurate predictions (as indicated by the iid results).

Results for window size 3 are similar. However, the improvement of i-SECOC over iid and of c-SECOC over i-SECOC are smaller. This is expected since the larger window size spans multiple sequence positions, allowing the model to capture transition information using the observations alone, making the need for an explicit transition model less important. Nevertheless, both SECOC methods can capture useful transition structure that iid cannot, with c-SECOC benefiting from the use of cascade features. For both window sizes, we see that c-SECOC performs best for a particular cascade history length, and increasing beyond that length decreases accuracy by a small amount. This indicates that c-SECOC can suffer from overfitting as the cascade history grows.

Figures 5.2a and 5.2b show corresponding results for VP. We still observe that in-



(a) window size 1 with VP



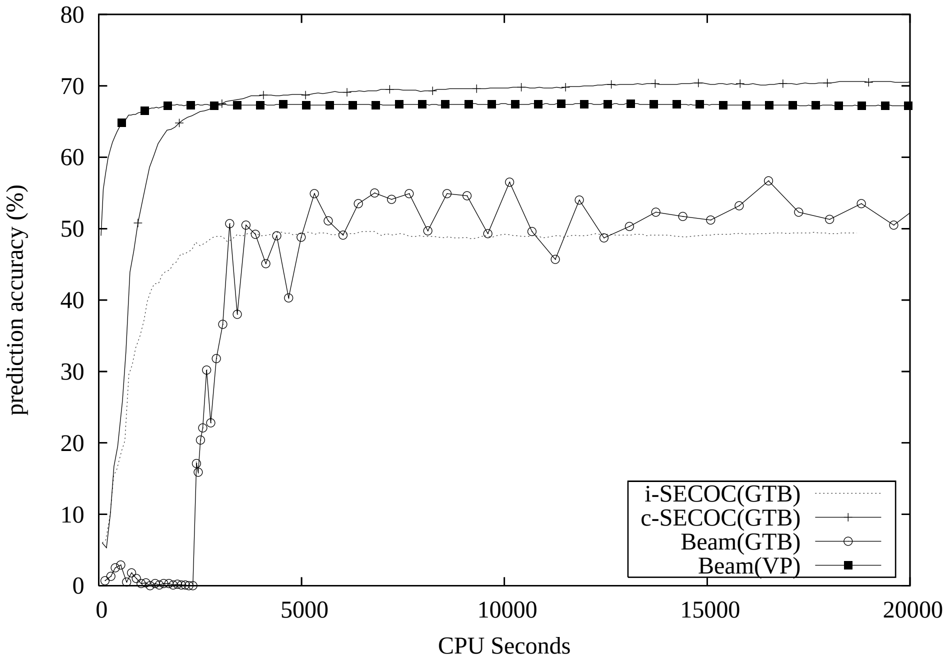
(b) window size 3 with VP

Figure 5.2: Nettalk data set with window sizes 1 and 3 trained by VP.

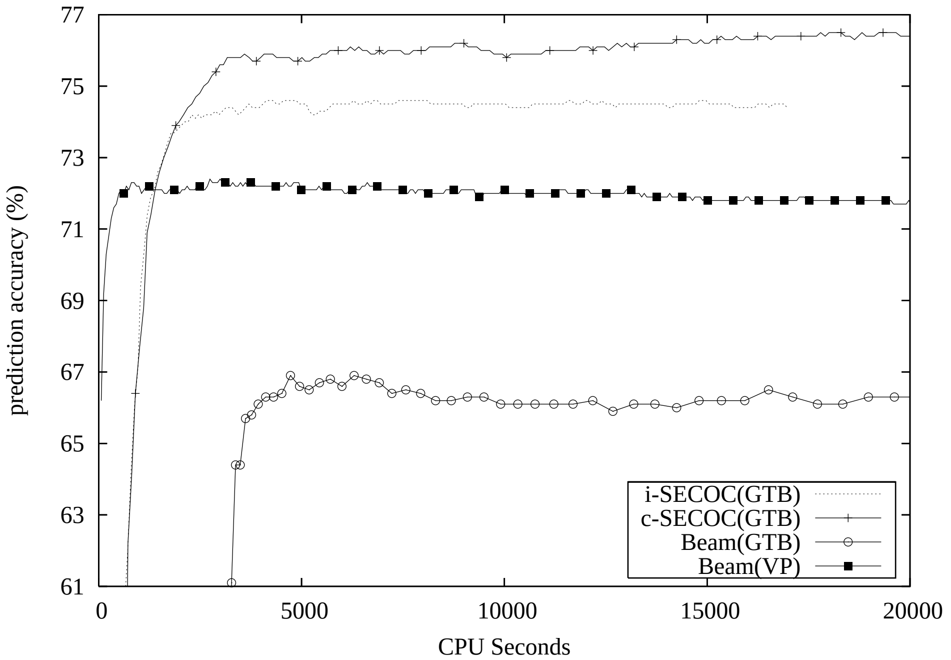
cluding cascade features allows c-SECOC to improve upon i-SECOC, though compared to GTB, longer cascade histories are required to achieve improvement. No overfitting was observed up to cascade history length 50. However, we were able to observe overfitting after code length 160 by including all possible cascade history bits, denoted by c-SECOC(all). All of our overfitting results suggest that in practice a limited validation process should be used to select the cascade history for c-SECOC. For large label sets, trying a small number of cascade history lengths is a much more practical alternative than training on the full label set.

5.2.2 GTB versus VP

c-SECOC using GTB performs significantly better than using VP. We explain this by noting that the critical feature of c-SECOC is that the binary CRFs are able to exploit the cascade features in order to better capture the transition structure. VP considers only linear combinations of these features, whereas GTB is able to capture non-linear relationships by inducing complex combinations of these features and hence capturing richer transition structure. This indicates that when using c-SECOC it can be important to use training methods such as GTB that are able to capture rich patterns in the observations and hence of the cascade features.



(a) window size 1



(b) window size 3

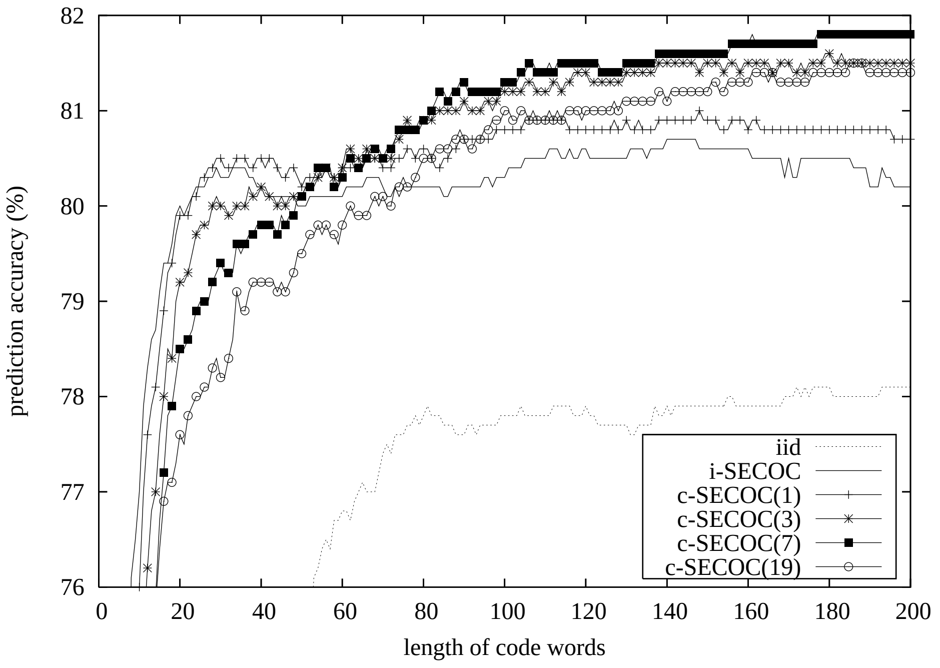
Figure 5.3: Nettalk: comparison between SECOC and beam search.

5.2.3 Comparing to Beam Search

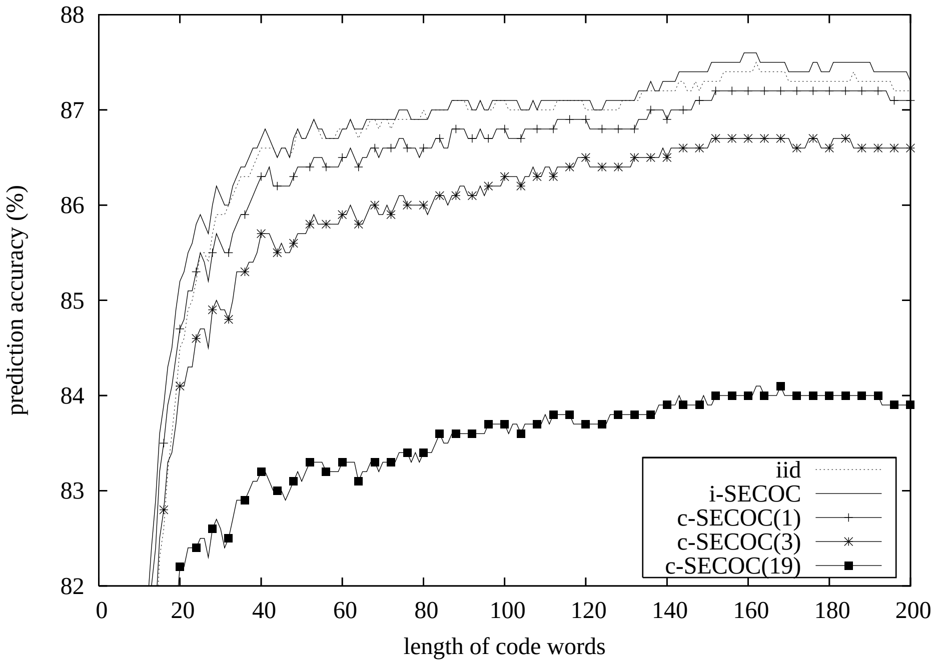
Here we compare the performance of c-SECOC with multi-label CRF models trained using beam search in place of full-Viterbi and forward-backward inference, which is a common approach to achieving practical inference with large label sets. For beam search, we tried various beam-sizes within reasonable computational limits. Figure 5.3 shows the accuracy/training-time trade-offs for the best beam-search results and the best results achieved for SECOC with 200 code-word bits and various cascade histories. The graph shows the test set performance versus the amount of training time in CPU seconds. First notice that GTB with beam search is significantly worse than for VP. We believe this is because GTB requires forward-backward inference during training whereas VP does not, and this is more adversely affected by beam search. Rather, c-SECOC using GTB performs significantly better than VP with beam search.

5.3 Noun Phrase Chunking

We consider the CoNLL 2000 Noun Phrase Chunking (NPC) shared task that involves labeling the words of sentences. This was one of the problems used in the original evaluation of i-SECOC. There are 121 class labels, which are combinations of Part-of-speech tagging labels and NPC labels, and 21,589 observation features for each word in the sequences. There are 8,936 sequences in the training set and 2,012 sequences in the test set. Due to the large number of observation features, we can not get good results for GTB using our current implementation and only present results for VP.



(a) window size 1



(b) window size 3

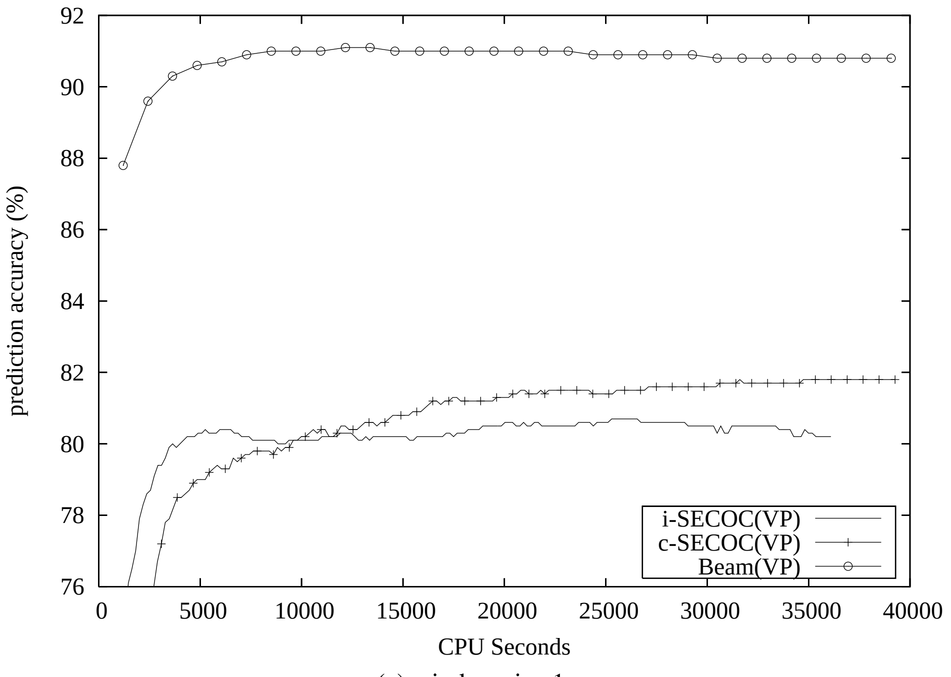
Figure 5.4: NPC data set with window sizes 1 and 3 trained by VP.

5.3.1 Comparing to i-SECOC

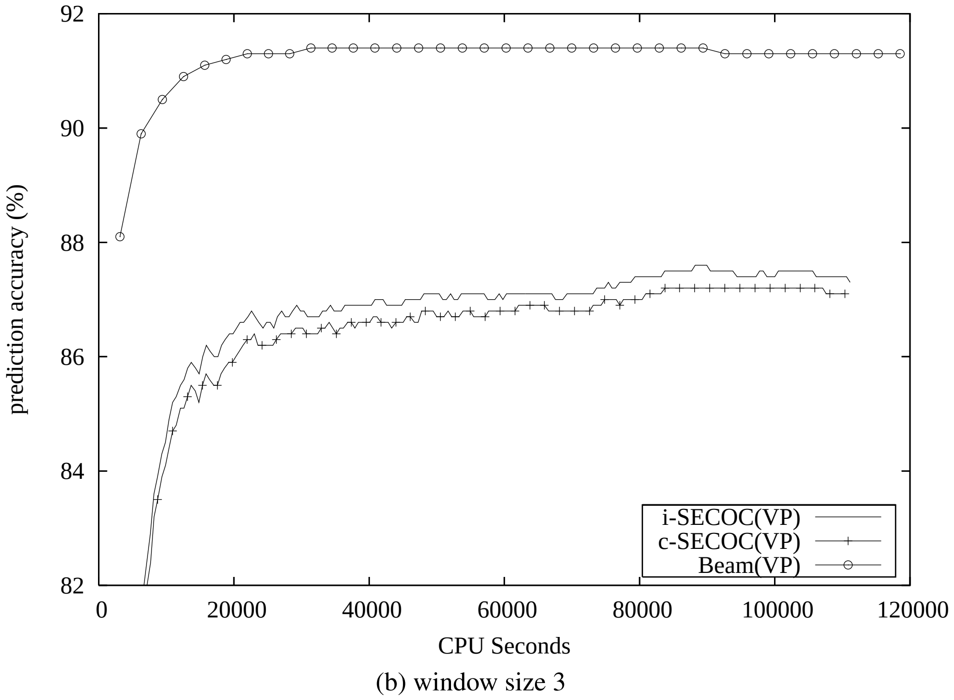
As shown in Figure 5.4a, for window size 1, i-SECOC outperforms iid and incorporating cascade features allows c-SECOC to outperform i-SECOC by a small margin. Again we see overfitting for c-SECOC for larger numbers of cascade features. Moving to window size 3 changes the story. Here a large amount of observation information is included from the current and adjacent sentence positions, and as a result iid performs as well as any of the SECOC approaches. The large amount of observation information at each sequence position appears to capture enough transition information so that SECOC gets little benefit from learning an explicit transition model. This suggests that the performance of the SECOC methods in this domain is primarily a reflection of the ability of non-sequential ECOC. This is interesting given the i-SECOC results in Cohn et al. (2005), where all domains involved a large amount of local observation information. IID results were not reported there.

5.3.2 Comparing to Beam Search

We compare the accuracy versus training-time for models trained with SECOC and with beam search, using the already described experimental setup. Figure 5.5 shows that beam search performs much better than the c-SECOC methods within the same training time using VP as the base learning algorithm. We believe the poor performance of c-SECOC compared to beam search is that using VP does not allow for rich patterns to be captured in the observations which include the cascade features. We hypothesize, as observed in Nettalk, that c-SECOC would perform as well or better than beam search



(a) window size 1



(b) window size 3

Figure 5.5: NPC: comparison between SECOC and beam search using VP.

given a base CRF method that can capture complex patterns in the observations.

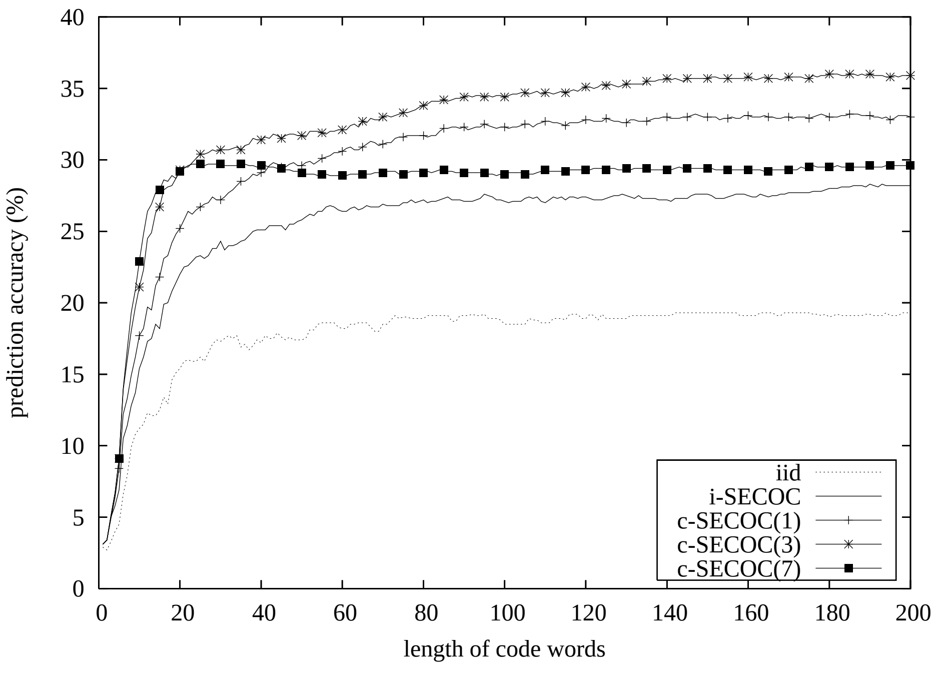
5.4 Synthetic Data Sets

Our results suggest that i-SECOC does poorly when critical transition structure is not captured by the observation features and that c-SECOC can improve on i-SECOC in such cases. Here we further evaluate these hypotheses.

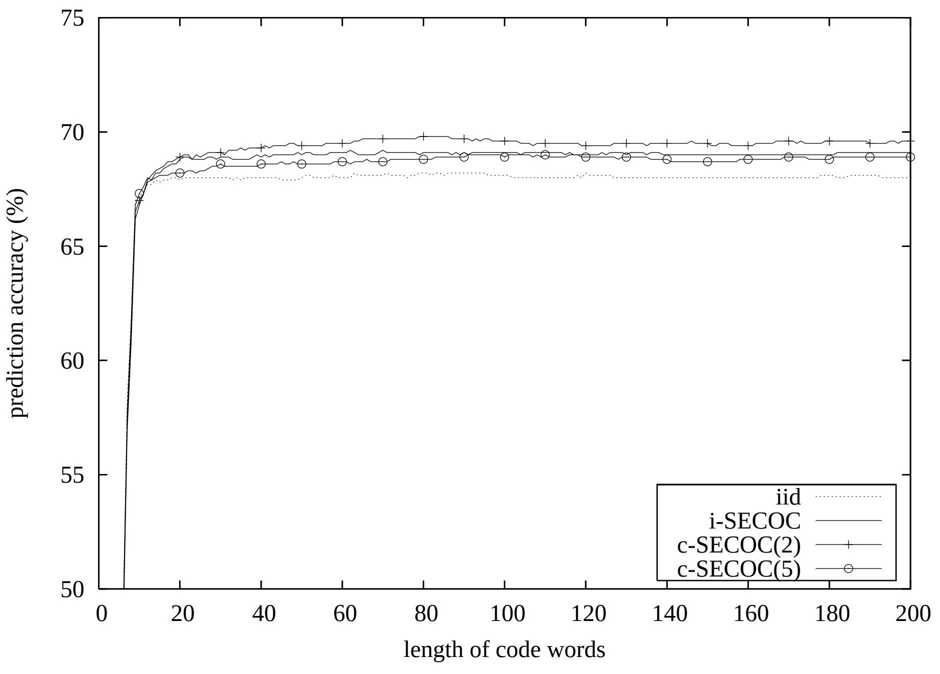
We generated data using a hidden Markov model (HMM) with 40 labels/states $\{l_1, \ldots, l_{40}\}$ and 40 possible observations $\{o_1, \ldots, o_{40}\}$. To specify the observation distribution, for each label l_i we randomly draw a set O_i of k_o observations not including o_i. If the current state is l_i, then the HMM generates the observation o_i with probability p_o and otherwise generates a randomly selected observation from O_i. The observation model is made more important by increasing p_o and decreasing k_o. The transition model is defined in a similar way. For each label l_i we randomly draw a set L_i of k_l labels not including l_i. If the current state is l_i then the HMM sets the next state to l_i with probability p_l and otherwise generates a random next state from L_i. Increasing p_l and decreasing k_l increases the transition model importance.

5.4.1 "Transition" Data Set

Here we use $p_o = 0.2$, $k_o = 8$, $p_l = 0.6$, $k_l = 2$, so that the transition structure is very important and observation features are quite uninformative. Figure 5.6a shows the results for GTB training using window size 1. We see that i-SECOC is significantly



(a) window size 1 on “transition” data set



(b) window size 3 on “both” data set

Figure 5.6: Synthetic data sets trained by GTB.

better than iid indicating that it is able to exploit transition structure not available to iid. However, including cascade features allows c-SECOC to further improve performance, showing that i-SECOC was unable to fully exploit information in the transition model. As in our previous experiments we observe that c-SECOC does overfit for the largest number of cascade features. For window size 3 (graph not shown) the results are similar except that the relative improvements are reduced since more observation information is available, making the transition model less critical. These results mirror those for the Nettalk data set.

5.4.2 “Both” Data Set

Here we use $p_o = 0.6$, $k_o = 2$, $p_l = 0.6$, $k_l = 2$ so that both the transition structure and observation features are very informative. Figure 5.6b shows results for GTB with window size 3. The observation features provide a large amount of information and performance of iid is similar to that of the SECOC variants. Also we see that c-SECOC is unable to improve on i-SECOC in this case. This suggests that the observation information captures the bulk of the transition information, and the performance of the SECOC methods is a reflection of non-sequential ECOC, rather than of their ability to explicitly capture transition structure.

Chapter 6 – Summary and Future Work

In this chapter, we briefly review the research work that has been done in this thesis and summarize our contributions. We also outline some research directions that deserve further exploration.

6.1 Summary

In this thesis, we focus on sequential supervised learning problems where the number of class labels is very large. Markovian models are commonly used to solve sequential supervised learning problems. However, even in first-order Markovian models, the time complexity of standard inference algorithms, such as the Viterbi algorithm and the forward-backward algorithm, scale quadratically with the number of class labels. When this number is very large, a number of recent methods for training Markovian models will become very expensive, because in those training methods inference is performed repeatedly.

The recent SECOC algorithm successfully reduces the training cost of a multi-label CRF by converting it into a set of binary-label CRFs based on the idea of error-correcting output coding. However, we uncovered empirical and theoretical shortcomings of independently trained SECOC. Independent training of binary-label CRFs can perform poorly when it is critical to explicitly capture complex transition models. In this thesis, we proposed cascaded SECOC and showed that it can improve accuracy in such situations. We also showed that when using less powerful CRF base learning algorithms, approaches other than SECOC (e.g. beam search) may be preferable.

6.2 Future Work

Within the research direction of this thesis, there are several open problems that need to be further explored. The first problem is how to design efficient validation procedures for selecting cascade history lengths. The second problem is that instead of generating a fixed code matrix beforehand, how can we incrementally generate a code matrix in a column-wise style based on the performance of the current code matrix. It is also desirable to provide a wide comparison of methods for dealing with large label sets in sequential supervised learning problems.

Bibliography

Leo Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth Publishing Company, Belmont, California, U.S.A., 1984.

Trevor Cohn, Andrew Smith, and Miles Osborne. Scaling conditional random fields using error correcting codes. In *Annual Meeting of the Association for Computational Linguistics*, pages 10–17, 2005.

Michael Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1–8, Philadelphia, July 2002. Association for Computational Linguistics.

Michael Collins and Brian Roark. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 111–118, 2004.

Thomas G. Dietterich. Machine learning for sequential data: A review. In *Proceedings of the Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition*, pages 15–30, London, UK, 2002. Springer-Verlag.

Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

Thomas G. Dietterich, Adam Ashenfelter, and Yaroslav Bulatov. Training conditional random fields via gradient tree boosting. In *Proceedings of the 21th Annual International Conference on Machine Learning (ICML 2004)*. ACM, 2004.

P. Felzenszwalb, D. Huttenlocher, and J. Kleinberg. Fast algorithms for large-state-space HMMs with applications to web usage analysis. In *NIPS 16*, 2003.

Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

Stuart Geman and Donald Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, Nov. 1984.

John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning (ICML 2001)*, pages 282–289. Morgan Kaufmann, 2001.

John Langford and Alina Beygelzimer. Sensitive error correcting output codes. In *COLT*, 2005.

Andrew McCallum, Dayne Freitag, and Fernando C. N. Pereira. Maximum entropy Markov models for information extraction and segmentation. In *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pages 591–598. Morgan Kaufmann, 2000.

Chris Pal, Charles Sutton, and Andrew McCallum. Sparse forward-backward using minimum divergence beams for fast training of conditional random fields. In *International Conference on Acoustics, Speech, and Signal Processing*, 2006.

Ning Qian and Terrence J. Sejnowski. Predicting the secondary structure of globular proteins using neural network models. *Journal of Molecular Biology*, 202:865–884, 1988.

Adwait Ratnaparkhi. A maximum entropy model for part-of-speech tagging. In Eric Brill and Kenneth Church, editors, *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 133–142, Somerset, New Jersey, 1996. Association for Computational Linguistics.

Terrence J. Sejnowski and Charles R. Rosenberg. Parallel networks that learn to pronounce English text. *Complex Systems*, 1:145–168, 1987.

Fei Sha and Fernando Pereira. Shallow parsing with conditional random fields. In Marti Hearst and Mari Ostendorf, editors, *HLT-NAACL 2003: Main Proceedings*, pages 213–220, Edmonton, Alberta, Canada, May 27 – June 1 2003. Association for Computational Linguistics.

Sajid Siddiqi and Andrew Moore. Fast inference and learning in large-state-space HMMs. In *ICML*, 2005.

Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin Markov networks. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 25–32. MIT Press, Cambridge, MA, 2004.

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the 21 st International Conference on Machine Learning*, Banff, Canada, 2004.

APPENDICES

Appendix A – Proof of the Stationary Distribution in Counter Example

Assume the transition probabilities are given as

$$\begin{aligned} P(y_t = 1 | y_{t-1} = 0) &= 0.5 \\ P(y_t = 0 | y_{t-1} = 0) &= 0.5 \\ P(y_t = 1 | y_{t-1} = 1) &= 0 \\ P(y_t = 0 | y_{t-1} = 1) &= 1. \end{aligned}$$

The Stationary distribution $P(y = 1)$ can be computed as follows.

$$\begin{aligned} P(y_t = 1) &= P(y_t = 1 | y_{t-1} = 1) \cdot P(y_{t-1} = 1) + \\ &\quad P(y_t = 1 | y_{t-1} = 0) \cdot P(y_{t-1} = 0) \\ &= 0 \cdot P(y_{t-1} = 1) + 0.5 \cdot P(y_{t-1} = 0) \\ &= 0.5 \cdot [1 - P(y_{t-1} = 1)] \end{aligned}$$

So we have

$$\begin{aligned} P(y = 1) &= 0.5 \cdot [1 - P(y = 1)] \\ P(y = 1) &= 1/3 \end{aligned}$$