Revisiting Output Coding for Sequential Supervised Learning

Guohua Hao and **Alan Fern**
School of Electrical Engineering and Computer Science
Oregon State University
Corvallis, OR 97331 USA
{haog, afern}@eecs.oregonstate.edu

Abstract

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.

1 Introduction

Sequence labeling is the problem of assigning a class label to each element of an input sequence. We study supervised learning for sequence-labeling problems where the number of labels is large. Markov models are widely used for sequence labeling, however, the time complexity of standard inference algorithms, such as Viterbi and forward-backward, scales quadratically with the number of labels. When this 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; Collins, 2002], repeatedly perform inference during learning. As the number of labels grows, such approaches quickly become computationally impractical.

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—e.g. 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 sequence-labeling problems that satisfy the assumptions of the models.

A third recent approach, which is the focus of this paper, 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.

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 paper is to show that this result does not hold for sequential learning. That is, there are sequence labeling 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 paper are: 1) to better understand the types of problems for which SECOC is suited, 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 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.

2 Conditional Random Fields

We study the sequence-labeling problem, where the goal is to learn a classifier that when given an observation sequence $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)$ returns a label sequence $Y = (y_1, y_2, \ldots, y_T)$ that assigns the correct class label to each observation. In this work, we will use conditional random fields (CRFs) for sequence labeling. A (sequential) CRF is a Markov random field [Geman and Geman, 1984] over the label sequence Y globally conditioned on the observation sequence $\mathbf{X}$. The Hammersley-Clifford theorem states that the conditional distribution $P(Y|\mathbf{X})$ has the following form:

$$P(Y|\mathbf{X}) = \frac{1}{Z(\mathbf{X})} \exp \sum_t \Psi(y_{t-1}, y_t, \mathbf{X}),$$

where $Z(\mathbf{X})$ is a normalization constant, and $\Psi(y_{t-1}, y_t, \mathbf{X})$ is the potential function defined on the clique (y_{t-1}, y_t). For simplicity, we only consider the pair-wise cliques (y_{t-1}, y_t), i. e. chain-structured CRFs, and assume that the potential function does not depend on position t. As in [Lafferty *et al.*, 2001], the potential function is usually represented as a linear combination of binary features:

$$\Psi(y_{t-1}, y_t, \mathbf{X}) = \sum_\alpha \lambda_\alpha f_\alpha(y_t, \mathbf{X}) + \sum_\beta \gamma_\beta g_\beta(y_t, y_{t-1}, \mathbf{X}).$$

The functions f capture the relationship between the label y_t and the input sequence $\mathbf{X}$; the boolean functions g capture the transitions between y_{t-1} and y_t (conditioned on $\mathbf{X}$).

Many CRF training algorithms have been proposed for training the parameters λ and γ, however, most all of these methods involve the repeated computation of the partition function $Z(\mathbf{X})$ 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. Below we describe the use of output coding for combating this computational burden.

3 ECOC for Sequence Labeling

For non-sequential supervised classification, the idea of Error-Correcting Output Coding (ECOC) has been successfully used to solve multi-class problems [Dietterich and Bakiri, 1995]. A multi-class learning problem with training examples $\{(\mathbf{x}_i, y_i)\}$ is reduced to a number of binary-class learning problems by assigning each class label j a binary vector, or code word, C_j of length n. A code matrix M is built by taking the code words as rows. Each column b_k in M can be regarded as a binary partition of the original label set, $b_k(y) \in \{0, 1\}$. We can then learn a binary classifier h_k using training examples $\{(\mathbf{x}_i, b_k(y_i))\}$. To predict the label of a new instance $\mathbf{x}$, we concatenate the predictions of each 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 $\hat{y} = \operatorname{argmin}_j \Delta(H(\mathbf{x}), C_j)$, where Δ is some distance measure, such as Hamming distance. In some implementations $H(\mathbf{x})$ stores probabilities rather than binary labels.

This idea has recently been applied to sequence labeling problems under the name sequential error-correcting output coding (SECOC) [Cohn *et al.*, 2005], 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 of a code matrix M. More specifically the training data for CRF h_k is given by $\{(\mathbf{X}_i, b_k(Y_i))\}$, where $b_k(Y) = (b_k(y_1), b_k(y_2), \ldots, b_k(y_T))$ is a binary label sequence. Given a set of binary classifiers for a code matrix and an observation sequence $\mathbf{X}$ we compute the multi-label output sequence as follows. We first use the forward-backward algorithm on each h_k to compute the probability that each sequence position should be labeled 1. For each sequence element $\mathbf{x}_t$ we then form a probability vector $H(\mathbf{x}_t) = (p_1, ..., p_n)$ where p_k is the probability computed by h_k for $\mathbf{x}_t$. We then let the label for $\mathbf{x}_t$ be the class label y_t whose codeword is closest to $H(\mathbf{x}_t)$ based on L_1 distance. The complexity of this inference process 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.

Prior work [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 one another 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.

SECOC Counter Example. 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 is sufficient for capturing all non-trivial codewords for this label set—i. e. 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 i.i.d. predictions according to the stationary distribution that predicts 1 with probability 1/3. 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.

4 Cascaded SECOC Training of CRFs

Each binary CRF in SECOC has very limited knowledge about the Markovian transition structure. One possibility to improve this situation is to provide limited coupling between the binary CRFs. One way to do this is to include observation features in each binary CRF that record the binary predictions of previous CRFs. We call this approach cascaded SECOC (c-SECOC), as opposed to independent SECOC (i-SECOC).

Given training examples $S = \{(\mathbf{X}_i, Y_i)\}$, let $Y_i^{(j)}$ be the prediction of the binary CRF learned with the j-th binary partition b_j and $y_{it}^{(j)}$ be the t-th element of $Y_i^{(j)}$. To train the binary CRF h_k based on binary partition b_k, each training example $(\mathbf{X}_i, Y_i)$ is extended to $(\mathbf{X}'_i, b_k(Y_i))$, 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:

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

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 Y for a given observation sequence $\mathbf{X}$, 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 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 section.

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 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, i. e. 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.

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.

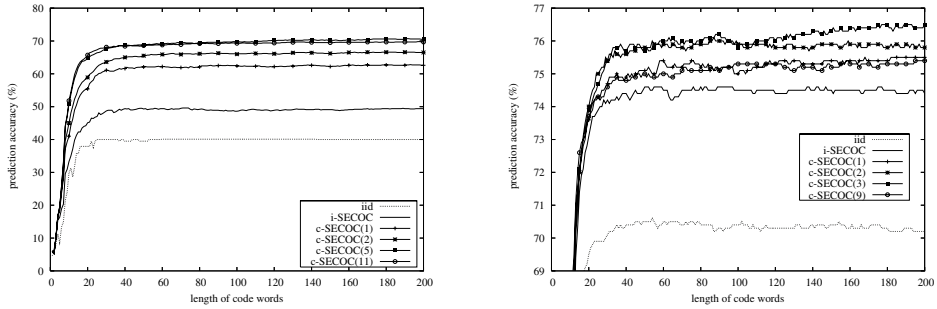
5.1 Nettalk Data Set

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.

Comparing to i-SECOC. Figures 1a and 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).

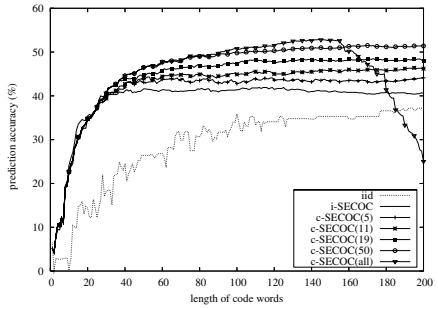
Results for window size 3 are similar. However, the improvement of i-SECOC over iid and of c-SECOC over i-SECOC are smaller. This is expected since the larger window size spans multiple sequence positions, allowing the model to capture transition information using the observations alone, making the need 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 1c and 1d show corresponding results for VP. We still observe that including cascade features allows c-SECOC

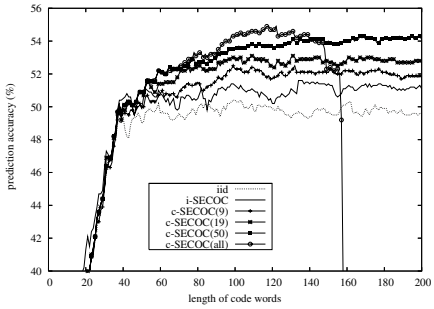


(a) window size 1 with GTB

(b) window size 3 with GTB



(c) window size 1 with VP



(d) window size 3 with VP

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

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.

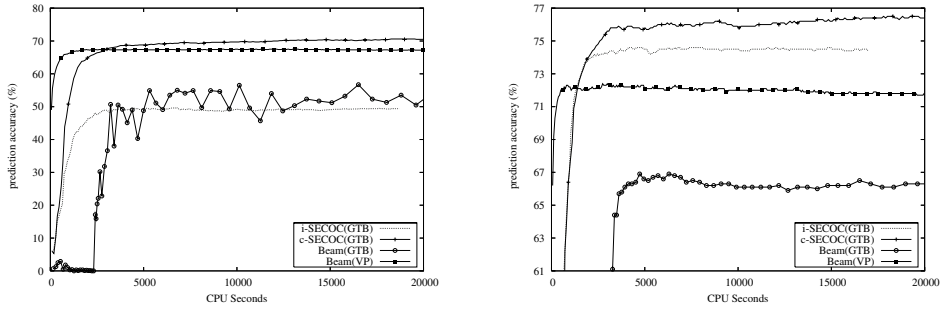
GTB versus VP. c-SECOC using GTB performs significantly better than using VP. We explain this by noting that the critical feature of c-SECOC is that the binary CRFs are able exploit the cascade features in order to better capture the transition structure. VP considers only linear combinations of these 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.

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 2 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.

5.2 Noun Phrase Chunking

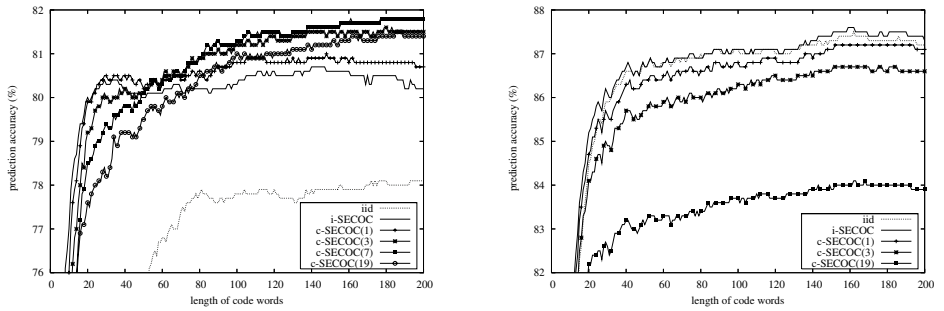
We consider the CoNLL 2000 Noun Phrase Chunking (NPC) shared task that involves labeling the words of sentences. This was one of the problems used in the original evaluation of i-SECOC. There are 121 class labels, which are combinations of Part-of-speech tagging labels and NPC labels, and 21,589 observation 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.

Comparing to i-SECOC. As shown in Figure 3a, 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 the 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 are 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.



(a) window size 1 (b) window size 3

Figure 2: Nettalk: comparison between SECOC and beam search.



(a) window size 1 (b) window size 3

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

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

5.3 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.

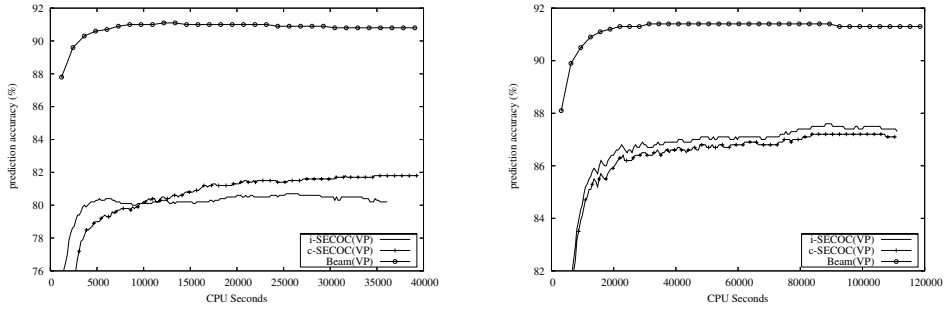
We generated data using a hidden Markov model (HMM) with 40 labels/states $\{l_1, \ldots, l_{40}\}$ and 40 possible observations $\{o_1, \ldots, o_{40}\}$. To specify the observation distribution, for each label l_i we randomly draw a set O_i of k_o observations not including o_i. If the current state 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.

"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 5a shows the results for GTB training using window size 1. We see that i-SECOC is significantly 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 less since more observation information is available, making the transition model less critical. These results mirror those for the Nettalk data set.

"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 5b 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.

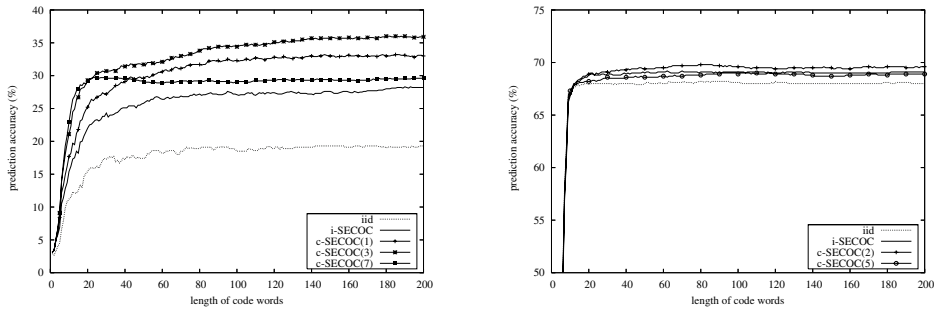
6 Summary

We uncovered empirical and theoretical shortcomings of independently trained SECOC. Independent training of binary



(a) window size 1 (b) window size 3

Figure 4: NPC: comparison between SECOC and beam search using VP.



(a) window size 1 on "transition" data set (b) window size 3 on "both" data set

Figure 5: Synthetic data sets trained by GTB.

CRFs can perform poorly when it is critical to explicitly capture complex transition models. 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. Future directions include efficient validation procedures for selecting cascade history lengths, incremental generation of code words, and a wide comparison of methods for dealing with large label sets.

Acknowledgments

We thank John Langford for discussion of the counter example to independent SECOC and Thomas Dietterich for his support. This work was supported by NSF grant IIS-0307592.

References

[Cohn *et al.*, 2005] 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.

[Collins and Roark, 2004] 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.

[Collins, 2002] M. Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Empirical Methods in NLP*, pages 1–8, 2002.

[Dietterich and Bakiri, 1995] Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *JAIR*, 2:263–286, 1995.

[Dietterich *et al.*, 2004] Thomas G. Dietterich, Adam Ashenfelter, and Yaroslav Bulatov. Training conditional random fields via gradient tree boosting. In *ICML*, 2004.

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

[Geman and Geman, 1984] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *PAMI*, 6:721–741, 1984.

[Lafferty *et al.*, 2001] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289, 2001.

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

[Pal *et al.*, 2006] 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.

[Sejnowski and Rosenberg, 1987] T. J. Sejnowski and C. R. Rosenberg. Parallel networks that learn to pronouce English text. *Complex Systems*, 1:145–168, 1987.

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

[Tsochantaridis *et al.*, 2004] Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. Support vector machine learning for interdependent and structured output spaces. In *ICML*, 2004.