Scaling Conditional Random Fields Using Error-Correcting Codes

Trevor Cohn
Department of Computer Science and Software Engineering
University of Melbourne, Australia
tacohn@csse.unimelb.edu.au

Andrew Smith
Division of Informatics
University of Edinburgh
United Kingdom
a.p.smith-2@sms.ed.ac.uk

Miles Osborne
Division of Informatics
University of Edinburgh
United Kingdom
miles@inf.ed.ac.uk

Abstract

Conditional Random Fields (CRFs) have been applied with considerable success to a number of natural language processing tasks. However, these tasks have mostly involved very small label sets. When deployed on tasks with larger label sets, the requirements for computational resources mean that training becomes intractable.

This paper describes a method for training CRFs on such tasks, using error correcting output codes (ECOC). A number of CRFs are independently trained on the separate binary labelling tasks of distinguishing between a subset of the labels and its complement. During decoding, these models are combined to produce a predicted label sequence which is resilient to errors by individual models.

Error-correcting CRF training is much less resource intensive and has a much faster training time than a standardly formulated CRF, while decoding performance remains quite comparable. This allows us to scale CRFs to previously impossible tasks, as demonstrated by our experiments with large label sets.

1 Introduction

Conditional random fields (CRFs) (Lafferty et al., 2001) are probabilistic models for labelling sequential data. CRFs are undirected graphical models that define a conditional distribution over label sequences given an observation sequence. They allow the use of arbitrary, overlapping, non-independent features as a result of their global conditioning. This allows us to avoid making unwarranted independence assumptions over the observation sequence, such as those required by typical generative models.

Efficient inference and training methods exist when the graphical structure of the model forms a chain, where each position in a sequence is connected to its adjacent positions. CRFs have been applied with impressive empirical results to the tasks of named entity recognition (McCallum and Li, 2003), simplified part-of-speech (POS) tagging (Lafferty et al., 2001), noun phrase chunking (Sha and Pereira, 2003) and extraction of tabular data (Pinto et al., 2003), among other tasks.

CRFs are usually estimated using gradient-based methods such as limited memory variable metric (LMVM). However, even with these efficient methods, training can be slow. Consequently, most of the tasks to which CRFs have been applied are relatively small scale, having only a small number of training examples and small label sets. For much larger tasks, with hundreds of labels and millions of examples, current training methods prove intractable. Although training can potentially be parallelised and thus run more quickly on large clusters of computers, this in itself is not a solution to the problem: tasks can reasonably be expected to increase in size and complexity much faster than any increase in computing power. In order to provide scalability, the factors which most affect the resource usage and runtime of the training method

Proceedings of the 43rd Annual Meeting of the ACL, pages 10–17,
Ann Arbor, June 2005.

must be addressed directly – ideally the dependence on the number of labels should be reduced.

This paper presents an approach which enables CRFs to be used on larger tasks, with a significant reduction in the time and resources needed for training. This reduction does not come at the cost of performance – the results obtained on benchmark natural language problems compare favourably, and sometimes exceed, the results produced from regular CRF training. *Error correcting output codes* (ECOC) (Dietterich and Bakiri, 1995) are used to train a community of CRFs on binary tasks, with each discriminating between a subset of the labels and its complement. Inference is performed by applying these 'weak' models to an unknown example, with each component model removing some ambiguity when predicting the label sequence. Given a sufficient number of binary models predicting suitably diverse label subsets, the label sequence can be inferred while being robust to a number of individual errors from the weak models. As each of these weak models are binary, individually they can be efficiently trained, even on large problems. The number of weak learners required to achieve good performance is shown to be relatively small on practical tasks, such that the overall complexity of error-correcting CRF training is found to be much less than that of regular CRF training methods.

We have evaluated the error-correcting CRF on the CoNLL 2003 named entity recognition (NER) task (Sang and Meulder, 2003), where we show that the method yields similar generalisation performance to standardly formulated CRFs, while requiring only a fraction of the resources, and no increase in training time. We have also shown how the error-correcting CRF scales when applied to the larger task of POS tagging the Penn Treebank and also the even larger task of simultaneously noun phrase chunking (NPC) and POS tagging using the CoNLL 2000 data-set (Sang and Buchholz, 2000).

2 Conditional random fields

CRFs are undirected graphical models used to specify the conditional probability of an assignment of output labels given a set of input observations. We consider only the case where the output labels of the model are connected by edges to form a linear chain. The joint distribution of the label sequence, $\mathbf{y}$, given the input observation sequence, $\mathbf{x}$, is given by

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp \sum_{t=1}^{T+1} \sum_{k} \lambda_k f_k(t, \mathbf{y}_{t-1}, \mathbf{y}_t, \mathbf{x})$$

where T is the length of both sequences and λ_k are the parameters of the model. The functions f_k are feature functions which map properties of the observation and the labelling into a scalar value. $Z(\mathbf{x})$ is the partition function which ensures that p is a probability distribution.

A number of algorithms can be used to find the optimal parameter values by maximising the log-likelihood of the training data. Assuming that the training sequences are drawn *IID* from the population, the conditional log likelihood $\mathcal{L}$ is given by

$$\begin{aligned} \mathcal{L} &= \sum_i \log p(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}) \\ &= \sum_i \left\{ \sum_{t=1}^{T^{(i)}+1} \sum_k \lambda_k f_k(t, \mathbf{y}_{t-1}^{(i)}, \mathbf{y}_t^{(i)}, \mathbf{x}^{(i)}) \right. \\ &- \left. \log Z(\mathbf{x}^{(i)}) \right\} \end{aligned}$$

where $\mathbf{x}^{(i)}$ and $\mathbf{y}^{(i)}$ are the i^{th} observation and label sequence. Note that a prior is often included in the $\mathcal{L}$ formulation; it has been excluded here for clarity of exposition. CRF estimation methods include generalised iterative scaling (GIS), improved iterative scaling (IIS) and a variety of gradient based methods. In recent empirical studies on maximum entropy models and CRFs, limited memory variable metric (LMVM) has proven to be the most efficient method (Malouf, 2002; Wallach, 2002); accordingly, we have used LMVM for CRF estimation.

Every iteration of LMVM training requires the computation of the log-likelihood and its derivative with respect to each parameter. The partition function $Z(\mathbf{x})$ can be calculated efficiently using dynamic programming with the forward algorithm. $Z(\mathbf{x})$ is given by $\sum_y \alpha_T(y)$ where α are the forward values, defined recursively as

$$\alpha_{t+1}(y) = \sum_{y'} \alpha_t(y') \exp \sum_k \lambda_k f_k(t+1, y', y, \mathbf{x})$$

The derivative of the log-likelihood is given by

$$\frac{\partial \mathcal{L}}{\partial \lambda_k} = \sum_i \left\{ \sum_{t=1}^{T^{(i)}+1} f_k(t, \mathbf{y}_{t-1}^{(i)}, \mathbf{y}_t^{(i)}, \mathbf{x}^{(i)}) - \sum_{\mathbf{y}} p(\mathbf{y}|\mathbf{x}^{(i)}) \sum_{t=1}^{T^{(i)}+1} f_k(t, \mathbf{y}_{t-1}, \mathbf{y}_t, \mathbf{x}^{(i)}) \right\}$$

The first term is the empirical count of feature k, and the second is the expected count of the feature under the model. When the derivative equals zero – at convergence – these two terms are equal. Evaluating the first term of the derivative is quite simple. However, the sum over all possible labellings in the second term poses more difficulties. This term can be factorised, yielding

$$\sum_t \sum_{y',y} p(Y_{t-1} = y', Y_t = y|\mathbf{x}^{(i)}) f_k(t, y', y, \mathbf{x}^{(i)})$$

This term uses the marginal distribution over pairs of labels, which can be efficiently computed from the forward and backward values as

$$\frac{\alpha_{t-1}(y') \exp \sum_k \lambda_k f_k(t, y', y, \mathbf{x}^{(i)}) \beta_t(y)}{Z(\mathbf{x}^{(i)})}$$

The backward probabilities β are defined by the recursive relation

$$\beta_t(y) = \sum_{y'} \beta_{t+1}(y') \exp \sum_k \lambda_k f_k(t+1, y, y', \mathbf{x})$$

Typically CRF training using LMVM requires many hundreds or thousands of iterations, each of which involves calculating of the log-likelihood and its derivative. The time complexity of a single iteration is $O(L^2NTF)$ where L is the number of labels, N is the number of sequences, T is the average length of the sequences, and F is the average number of activated features of each labelled clique. It is not currently possible to state precise bounds on the number of iterations required for certain problems; however, problems with a large number of sequences often require many more iterations to converge than problems with fewer sequences. Note that efficient CRF implementations cache the feature values for every possible clique labelling of the training data, which leads to a memory requirement with the same complexity of $O(L^2NTF)$ – quite demanding even for current computer hardware.

3 Error Correcting Output Codes

Since the time and space complexity of CRF estimation is dominated by the square of the number of labels, it follows that reducing the number of labels will significantly reduce the complexity. Error-correcting coding is an approach which recasts multiple label problems into a set of binary label problems, each of which is of lesser complexity than the full multiclass problem. Interestingly, training a set of binary CRF classifiers is overall much more efficient than training a full multi-label model. This is because error-correcting CRF training reduces the L^2 complexity term to a constant. Decoding proceeds by predicting these binary labels and then recovering the encoded actual label.

Error-correcting output codes have been used for text classification, as in Berger (1999), on which the following is based. Begin by assigning to each of the m labels a unique n-bit string $\mathcal{C}_i$, which we will call the *code* for this label. Now train n binary classifiers, one for each column of the coding matrix (constructed by taking the labels' *codes* as rows). The j^{th} classifier, γ^j, takes as positive instances those with label i where $\mathcal{C}_{ij} = 1$. In this way, each classifier learns a different concept, discriminating between different subsets of the labels.

We denote the set of binary classifiers as $\Gamma = \{\gamma^1, \gamma^2, \ldots, \gamma^n\}$, which can be used for prediction as follows. Classify a novel instance x with each of the binary classifiers, yielding a n-bit vector $\Gamma(x) = \{\gamma^1(x), \gamma^2(x), \ldots, \gamma^n(x)\}$. Now compare this vector to the codes for each label. The vector may not exactly match any of the labels due to errors in the individual classifiers, and thus we chose the actual label which minimises the distance $\text{argmin}_i \Delta(\Gamma(x), \mathcal{C}_i)$. Typically the Hamming distance is used, which simply measures the number of differing bit positions. In this manner, prediction is resilient to a number of prediction errors by the binary classifiers, provided the codes for the labels are sufficiently diverse.

3.1 Error-correcting CRF training

Error-correcting codes can also be applied to sequence labellers, such as CRFs, which are capable of multiclass labelling. ECOCs can be used with CRFs in a similar manner to that given above for

classifiers. A series of CRFs are trained, each on a relabelled variant of the training data. The relabelling for each binary CRF maps the labels into binary space using the relevant column of the coding matrix, such that label i is taken as a positive for the j^{th} model example if $\mathcal{C}_{ij} = 1$.

Training with a binary label set reduces the time and space complexity for each training iteration to $O(NTF)$; the L^2 term is now a constant. Provided the code is relatively short (i.e. there are few binary models, or *weak* learners), this translates into considerable time and space savings. Coding theory doesn't offer any insights into the optimal code length (i.e. the number of weak learners). When using a very short code, the error-correcting CRF will not adequately model the decision boundaries between all classes. However, using a long code will lead to a higher degree of dependency between pairs of classifiers, where both model similar concepts. The generalisation performance should improve quickly as the number of weak learners (code length) increases, but these gains will diminish as the inter-classifier dependence increases.

3.2 Error-correcting CRF decoding

While training of error-correcting CRFs is simply a logical extension of the ECOC classifier method to sequence labellers, decoding is a different matter. We have applied three decoding different strategies. The **Standalone** method requires each binary CRF to find the Viterbi path for a given sequence, yielding a string of 0s and 1s for each model. For each position t in the sequence, the t^{th} bit from each model is taken, and the resultant bit string compared to each of the label codes. The label with the minimum Hamming distance is then chosen as the predicted label for that site. This method allows for error correction to occur at each site, however it discards information about the uncertainty of each weak learner, instead only considering the most probable paths.

The **Marginals** method of decoding uses the marginal probability distribution at each position in the sequence instead of the Viterbi paths. This distribution is easily computed using the forward backward algorithm. The decoding proceeds as before, however instead of a bit string we have a vector of probabilities. This vector is compared to each of the label codes using the L_1 distance, and the closest label is chosen. While this method incorporates the uncertainty of the binary models, it does so at the expense of the path information in the sequence.

Neither of these decoding methods allow the models to interact, although each individual weak learner may benefit from the predictions of the other weak learners. The **Product** decoding method addresses this problem. It treats each weak model as an independent predictor of the label sequence, such that the probability of the label sequence given the observations can be re-expressed as the product of the probabilities assigned by each weak model. A given labelling $\mathbf{y}$ is projected into a bit string for each weak learner, such that the i^{th} entry in the string is $\mathcal{C}_{kj}$ for the j^{th} weak learner, where k is the index of label $\mathbf{y}_i$. The weak learners can then estimate the probability of the bit string; these are then combined into a global product to give the probability of the label sequence

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z'(\mathbf{x})} \prod_j p_j(b_j(\mathbf{y})|\mathbf{x})$$

where $p_j(\mathbf{q}|\mathbf{x})$ is the predicted probability of $\mathbf{q}$ given $\mathbf{x}$ by the j^{th} weak learner, $b_j(\mathbf{y})$ is the bit string representing $\mathbf{y}$ for the j^{th} weak learner and $Z'(\mathbf{x})$ is the partition function. The log probability is

$$\sum_j \{F_j(b_j(\mathbf{y}), \mathbf{x}) \cdot \lambda_\mathbf{j} - \log Z_j(\mathbf{x})\} - \log Z'(\mathbf{x})$$

where $F_j(\mathbf{y}, \mathbf{x}) = \sum_{t=1}^{T+1} \mathbf{f}_j(t, \mathbf{y_{t-1}}, \mathbf{y_t}, \mathbf{x})$. This log probability can then be maximised using the Viterbi algorithm as before, noting that the two $\log$ terms are constant with respect to y and thus need not be evaluated. Note that this decoding is an equivalent formulation to a uniformly weighted logarithmic opinion pool, as described in Smith et al. (2005).

Of the three decoding methods, **Standalone** has the lowest complexity, requiring only a binary Viterbi decoding for each weak learner. **Marginals** is slightly more complex, requiring the forward and backward values. **Product**, however, requires Viterbi decoding with the full label set, and many features – the union of the features of each weak learner – which can be quite computationally demanding.

3.3 Choice of code

The accuracy of ECOC methods are highly dependent on the quality of the code. The ideal code has diverse rows, yielding a high error-correcting capability, and diverse columns such that the weak learners model highly independent concepts. When the number of labels, k, is small, an exhaustive code with every unique column is reasonable, given there are $2^{k-1} - 1$ unique columns. With larger label sets, columns must be selected with care to maximise the inter-row and inter-column separation. This can be done by randomly sampling the column space, in which case the probability of poor separation diminishes quickly as the number of columns increases (Berger, 1999). Algebraic codes, such as BCH codes, are an alternative coding scheme which can provide near-optimal error-correcting capability (MacWilliams and Sloane, 1977), however these codes provide no guarantee of good column separation.

4 Experiments

Our experiments show that error-correcting CRFs are highly accurate on benchmark problems with small label sets, as well as on larger problems with many more labels, which would be otherwise prove intractable for traditional CRFs. Moreover, with a good code, the time and resources required for training and decoding can be much less than that of the standardly formulated CRF.

4.1 Named entity recognition

CRFs have been used with strong results on the CoNLL 2003 NER task (McCallum, 2003) and thus this task is included here as a benchmark. This data set consists of a 14,987 training sentences (204,567 tokens) drawn from news articles, tagged for person, location, organisation and miscellaneous entities. There are 8 IOB-2 style labels.

A multiclass (standardly formulated) CRF was trained on these data using features covering word identity, word prefix and suffix, orthographic tests for digits, case and internal punctuation, word length, POS tag and POS tag bigrams before and after the current word. Only features seen at least once in the training data were included in the model, resulting in 450,345 binary features. The model was trained without regularisation and with a Gaussian prior. An exhaustive code was created with all 127 unique columns. All of the weak learners were trained with the same feature set, each having around 315,000 features. The performance of the standard and error-correcting models are shown in Table 1. We tested for statistical significance using the matched pairs test (Gillick and Cox, 1989) at $p < 0.001$. Those results which are significantly better than the corresponding multiclass MLE or regularised model are flagged with a *, and those which are significantly worse with a †.

Model	Decoding	MLE	Regularised
Multiclass		88.04	89.78
Coded	standalone	88.23*	88.67†
	marginals	88.23*	89.19
	product	88.69*	89.69

Table 1: F_1 scores on NER task.

These results show that error-correcting CRF training achieves quite similar performance to the multiclass CRF on the task (which incidentally exceeds McCallum (2003)'s result of 89.0 using feature induction). Product decoding was the better of the three methods, giving the best performance both with and without regularisation, although this difference was only statistically significant between the regularised standalone and the regularised product decoding. The unregularised error-correcting CRF significantly outperformed the multiclass CRF with all decoding strategies, suggesting that the method already provides some regularisation, or corrects some inherent bias in the model.

Using such a large number of weak learners is costly, in this case taking roughly ten times longer to train than the multiclass CRF. However, much shorter codes can also achieve similar results. The simplest code, where each weak learner predicts only a single label (a.k.a. one-vs-all), achieved an F score of 89.56, while only requiring 8 weak learners and less than half the training time as the multiclass CRF. This code has no error correcting capability, suggesting that the code's column separation (and thus interdependence between weak learners) is more important than its row separation.

An exhaustive code was used in this experiment simply for illustrative purposes: many columns in this code were unnecessary, yielding only a slight gain in performance over much simpler codes while incurring a very large increase in training time. Therefore, by selecting a good subset of the exhaustive code, it should be possible to reduce the training time while preserving the strong generalisation performance. One approach is to incorporate skew in the label distribution in our choice of code – the code should minimise the confusability of commonly occurring labels more so than that of rare labels. Assuming that errors made by the weak learners are independent, the probability of a single error, q, as a function of the code length n can be bounded by

$$q(n) \leq 1 - \sum_l p(l) \sum_{i=0}^{\lfloor \frac{h_l - 1}{2} \rfloor} \binom{n}{i} \hat{p}^i (1-\hat{p})^{n-i}$$

where $p(l)$ is the marginal probability of the label l, h_l is the minimum Hamming distance between l and any other label, and $\hat{p}$ is the maximum probability of an error by a weak learner. The performance achieved by selecting the code with the minimum loss bound from a large random sample of codes is shown in Figure 1, using standalone decoding, where $\hat{p}$ was estimated on the development set. For comparison, randomly sampled codes and a greedy oracle are shown. The two random sampled codes show those samples where no column is repeated, and where duplicate columns are permitted (random with replacement). The oracle repeatedly adds to the code the column which most improves its F_1 score. The minimum loss bound method allows the performance plateau to be reached more quickly than random sampling; i.e. shorter codes can be used, thus allowing more efficient training and decoding.

Note also that multiclass CRF training required 830Mb of memory, while error-correcting training required only 380Mb. Decoding of the test set (51,362 tokens) with the error-correcting model (exhaustive, MLE) took between 150 seconds for standalone decoding and 173 seconds for integrated decoding. The multiclass CRF was much faster, taking only 31 seconds, however this time difference could be reduced with suitable optimisations.

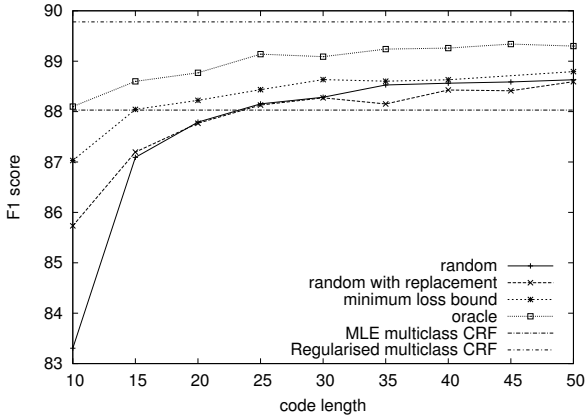

Figure 1: NER F1 scores for standalone decoding with random codes, a minimum loss code and a greedy oracle.

Coding	Decoding	MLE	Regularised
Multiclass		95.69	95.78
Coded - 200	standalone	95.63	96.03
	marginals	95.68	96.03
One-vs-all	product	94.90	96.57

Table 2: POS tagging accuracy.

4.2 Part-of-speech Tagging

CRFs have been applied to POS tagging, however only with a very simple feature set and small training sample (Lafferty et al., 2001). We used the Penn Treebank Wall Street Journal articles, training on sections 2–21 and testing on section 24. In this task there are 45,110 training sentences, a total of 1,023,863 tokens and 45 labels.

The features used included word identity, prefix and suffix, whether the word contains a number, uppercase letter or a hyphen, and the words one and two positions before and after the current word. A random code of 200 columns was used for this task. These results are shown in Table 2, along with those of a multiclass CRF and an alternative one-vs-all coding. As for the NER experiment, the decoding performance levelled off after 100 bits, beyond which the improvements from longer codes were only very slight. This is a very encouraging characteristic, as only a small number of weak learners are required for good performance.

The random code of 200 bits required 1,300Mb of RAM, taking a total of 293 hours to train and 3 hours to decode (54,397 tokens) on similar machines to those used before. We do not have figures regarding the resources used by Lafferty et al.'s CRF for the POS tagging task and our attempts to train a multiclass CRF for full-scale POS tagging were thwarted due to lack of sufficient available computing resources. Instead we trained on a 10,000 sentence subset of the training data, which required approximately 17Gb of RAM and 208 hours to train.

Our best result on the task was achieved using a one-vs-all code, which reduced the training time to 25 hours, as it only required training 45 binary models. This result exceeds Lafferty et al.'s accuracy of 95.73% using a CRF but falls short of Toutanova et al. (2003)'s state-of-the-art 97.24%. This is most probably due to our only using a first-order Markov model and a fairly simple feature set, where Tuotanova et al. include a richer set of features in a third order model.

4.3 Part-of-speech Tagging and Noun Phrase Segmentation

The joint task of simultaneously POS tagging and noun phrase chunking (NPC) was included in order to demonstrate the scalability of error-correcting CRFs. The data was taken from the CoNLL 2000 NPC shared task, with the model predicting both the chunk tags and the POS tags. The training corpus consisted of 8,936 sentences, with 47,377 tokens and 118 labels.

A 200-bit random code was used, with the following features: word identity within a window, prefix and suffix of the current word and the presence of a digit, hyphen or upper case letter in the current word. This resulted in about 420,000 features for each weak learner. A joint tagging accuracy of 90.78% was achieved using MLE training and standalone decoding. Despite the large increase in the number of labels in comparison to the earlier tasks, the performance also began to plateau at around 100 bits. This task required 220Mb of RAM and took a total of 30 minutes to train each of the 200 binary CRFs, this time on Pentium 4 machines with 1Gb RAM. Decoding of the 47,377 test tokens took 9,748 seconds and 9,870 seconds for the standalone and marginals methods respectively.

Sutton et al. (2004) applied a variant of the CRF, the dynamic CRF (DCRF), to the same task, modelling the data with two interconnected chains where one chain predicted NPC tags and the other POS tags. They achieved better performance and training times than our model; however, this is not a fair comparison, as the two approaches are orthogonal. Indeed, applying the error-correcting CRF algorithms to DCRF models could feasibly decrease the complexity of the DCRF, allowing the method to be applied to larger tasks with richer graphical structures and larger label sets.

In all three experiments, error-correcting CRFs have achieved consistently good generalisation performance. The number of weak learners required to achieve these results was shown to be relatively small, even for tasks with large label sets. The time and space requirements were lower than those of a traditional CRF for the larger tasks and, most importantly, did not increase substantially when the number of labels was increased.

5 Related work

Most recent work on improving CRF performance has focused on feature selection. McCallum (2003) describes a technique for greedily adding those feature conjuncts to a CRF which significantly improve the model's log-likelihood. His experimental results show that feature induction yields a large increase in performance, however our results show that standardly formulated CRFs can perform well above their reported 73.3%, casting doubt on the magnitude of the possible improvement. Roark et al. (2004) have also employed feature selection to the huge task of language modelling with a CRF, by partially training a voted perceptron then removing all features that the are ignored by the perceptron. The act of automatic feature selection can be quite time consuming in itself, while the performance and runtime gains are often modest. Even with a reduced number of features, tasks with a very large label space are likely to remain intractable.

6 Conclusion

Standard training methods for CRFs suffer greatly from their dependency on the number of labels, making tasks with large label sets either difficult or impossible. As CRFs are deployed more widely to tasks with larger label sets this problem will become more evident. The current 'solutions' to these scaling problems – namely feature selection, and the use of large clusters – don't address the heart of the problem: the dependence on the square of number of labels.

Error-correcting CRF training allows CRFs to be applied to larger problems and those with larger label sets than were previously possible, without requiring computationally demanding methods such as feature selection. On standard tasks we have shown that error-correcting CRFs provide comparable or better performance than the standardly formulated CRF, while requiring less time and space to train. Only a small number of weak learners were required to obtain good performance on the tasks with large label sets, demonstrating that the method provides efficient scalability to the CRF framework.

Error-correction codes could be applied to other sequence labelling methods, such as the voted perceptron (Roark et al., 2004). This may yield an increase in performance and efficiency of the method, as its runtime is also heavily dependent on the number of labels. We plan to apply error-correcting coding to dynamic CRFs, which should result in better modelling of naturally layered tasks, while increasing the efficiency and scalability of the method. We also plan to develop higher order CRFs, using error-correcting codes to curb the increase in complexity.

7 Acknowledgements

This work was supported in part by a PORES travelling scholarship from the University of Melbourne, allowing Trevor Cohn to travel to Edinburgh.

References

Adam Berger. 1999. Error-correcting output coding for text classification. In *Proceedings of IJCAI: Workshop on machine learning for information filtering*.

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

L. Gillick and Stephen Cox. 1989. Some statistical issues in the comparison of speech recognition algorithms. In *Proceedings of the IEEE Conference on Acoustics, Speech and Signal Processing*, pages 532–535, Glasgow, Scotland.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labelling sequence data. In *Proceedings of ICML 2001*, pages 282–289.

Florence MacWilliams and Neil Sloane. 1977. *The theory of error-correcting codes*. North Holland, Amsterdam.

Robert Malouf. 2002. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of CoNLL 2002*, pages 49–55.

Andrew McCallum and Wei Li. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of CoNLL 2003*, pages 188–191.

Andrew McCallum. 2003. Efficiently inducing features of conditional random fields. In *Proceedings of UAI 2003*, pages 403–410.

David Pinto, Andrew McCallum, Xing Wei, and Bruce Croft. 2003. Table extraction using conditional random fields. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 235–242.

Brian Roark, Murat Saraclar, Michael Collins, and Mark Johnson. 2004. Discriminative language modeling with conditional random fields and the perceptron algorithm. In *Proceedings of ACL 2004*, pages 48–55.

Erik F. Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of CoNLL 2000 and LLL 2000*, pages 127–132.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL 2003*, pages 142–147, Edmonton, Canada.

Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of HLT-NAACL 2003*, pages 213–220.

Andrew Smith, Trevor Cohn, and Miles Osborne. 2005. Logarithmic opinion pools for conditional random fields. In *Proceedings of ACL 2005*.

Charles Sutton, Khashayar Rohanimanesh, and Andrew McCallum. 2004. Dynamic conditional random fields: Factorized probabilistic models for labelling and segmenting sequence data. In *Proceedings of the ICML 2004*.

Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer. 2003. Feature rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of HLT-NAACL 2003*, pages 252–259.

Hanna Wallach. 2002. Efficient training of conditional random fields. Master's thesis, University of Edinburgh.