Comparisons of Sequence Labeling Algorithms and Extensions

Nam Nguyen NHNGUYEN@CS.CORNELL.EDU
Yunsong Guo GUOYS@CS.CORNELL.EDU
Department of Computer Science, Cornell University, Ithaca, NY 14853, USA

Abstract

In this paper, we survey the current state-of-art models for structured learning problems, including Hidden Markov Model (HMM), Conditional Random Fields (CRF), Averaged Perceptron (AP), Structured SVMs (SVM^{struct}), Max Margin Markov Networks (M^3N), and an integration of search and learning algorithm (SEARN). With all due tuning efforts of various parameters of each model, on the data sets we have applied the models to, we found that SVM^{struct} enjoys better performance compared with the others. In addition, we also propose a new method which we call the Structured Learning Ensemble (SLE) to combine these structured learning models. Empirical results show that our SLE algorithm provides more accurate solutions compared with the best results of the individual models.

1. Introduction

In recent years, the various structured learning problems have obtained much attention especially in the natural language processing community such as part-of-speech tagging and linguistic translation. The goal of structured learning tasks is to predict complex structures, such as sequences, strings, trees, lattices, or graphs. Due to the exponential size of the output space, structured learning problems tend to be more challenging than the conventional multi-class prediction problems. In attempting to better address these problems, many new algorithms have been proposed and the progress has been encouraging. Some of the better known methods include HMM (Rabiner, 1989), CRF (Lafferty et al., 2001), Perceptron (Collins, 2002), SVM^{struct} (Tsochantaridis et al., 2005), M^3N (Taskar et al., 2003) and SEARN (Daumé III et al., 2006). Although some limited comparisons were made, no previous work has systematically compared the performances of the above structured learning methods in a standardized setting. In this paper, we focus our study on the sequence labeling problems. We implemented or used available packages for each method to survey their performance. The two data sets we used are the part-of-speech (POS) tagging (The Penn Treebank, 2002) and the Optical Character Recognition (OCR) (Kassel, 1995). We also compared against $SVM^{multiclass}$ (Crammer & Singer, 2001), a non-structured learning method as a base-line method. In order to guarantee that each method is calibrated to provide the best possible performance, we devoted a considerable amount of effort to finding suitable parameters for each method. In our experiments, we found that SVM^{struct} produces the most accurate predictions in both tasks, but performances of other models are still comparable. Thus, we believe that a natural way to further improve the performance is to combine the predictions of all models.

In recent years, ensemble learning methods have shown promising results in multiclass classification problems (Caruana et al., 2004). We expect similar results when applying ensemble learning to structured learning problems. However, this is not a trivial task as it is not clear how to combine sequence predictions from different structured learning models. For example, in the sequence labeling task, a straightforward way to combine the sequence predictions is to apply majority voting on each token position independently. However, this majority combination does not take the advantage of the correlation of tokens in the same sequence. In this paper, we present Structured Learning Ensemble (SLE), a novel combination method for sequence predictions that actually incorporates correlations of label sequences. In addition, empirical results in section 4 show that SLE produces superior results when compared with the structured learning algorithms surveyed.

The paper is structured as follows: in section 2, we describe in detail all models that we evaluated; in section 3, we discuss how to combine the predictions of multiple models by SLE; in section 4, we show the results of various models including SLE; and we conclude the work in section 5.

Appearing in *Proceedings of the 24^{th} International Conference on Machine Learning*, Corvallis, OR, 2007.

2. Structured Learning Models

In sequence labeling problems, the output is a sequence of labels $\mathbf{y} = (y^1, ..., y^T)$ which corresponds to an observation sequence $\mathbf{x} = (x^1, ..., x^T)$. If each individual label can take values from set Σ, then the structured output problem can be considered as a multiclass classification problem with $|\Sigma|^T$ different classes.

For SVM^{struct}, M^{3}N, Perceptron and CRF, a feature map

$$\phi(\mathbf{x}, \mathbf{y}) = [\phi_1 \ \phi_2 \ \dots \ \phi_{|\Sigma|} \ \phi_{trans}]^\top \tag{1}$$

is utilized to learn a weight vector $\mathbf{w}$. In the above equation, ϕ_k=$\sum_{i=1}^n x_i \mathcal{I}(y_i = k) \ \forall k \in \{1, \dots, |\Sigma|\}$ and $\phi_{trans} = [c_{11} \ c_{12} \ \dots \ c_{TT}]^\top$ where c_{ij} is the number of observed transitions from the i^{th} alphabet to the j^{th} alphabet in Σ.

In the testing phase, the predicted sequence $\mathbf{y}^*$ is computed as:

$$\mathbf{y}^* = argmax_{\mathbf{y}} \mathbf{w}^\top \phi(\mathbf{x}, \mathbf{y}). \tag{2}$$

The $argmax$ above is solved efficiently using a Viterbi-like dynamic programming algorithm. In the following subsections, we give a brief description of how each model is trained. We also note the tuning parameter for each method.

Even though SVM^{struct}, M^{3}N, and SVM^{multiclass} are capable of incorporating many different kernel functions, for simplicity we only use a linear kernel for these learning algorithms

$$K(\phi(\mathbf{x}, \mathbf{y}), \phi(\mathbf{x}', \mathbf{y}')) \ = \ \langle \phi(\mathbf{x}, \mathbf{y}), \phi(\mathbf{x}', \mathbf{y}') \rangle. \tag{3}$$

2.1. SVM^{multiclass}

The multiclass SVM method described in (Crammer & Singer, 2001) serves as our baseline method. Instead of using the sequence pair $(\mathbf{x}, \mathbf{y})$ as a training example, $SVM^{multiclass}$ treats each token-label pair (x, y) in the sequence as a training example. Given a feature map $\phi(x, y) = [\phi_1 \ \phi_2 \ \dots \ \phi_{|\Sigma|}]^\top$ where $\phi_k = x\mathcal{I}(y = k)$, SVM^{multiclass} learns the weight vector $\mathbf{w}$ and slack variables ξ for the following quadratic optimization problem:

$$min_{\mathbf{w}, \xi} \quad \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{n}\sum_{i=1}^n \xi_i \tag{4}$$

$$s.t. \quad \forall i, \ \forall y \in \Sigma \setminus y_i :$$

$$\langle \mathbf{w}, (\phi(x_i, y_i) - \phi(x_i, y)) \rangle \geq 1 - \xi_i.$$

SVM^{multiclass} uses a cutting plane method to solve this optimization problem by iteratively adding the most violated constraint into the set of constraints being optimized for in the dual formulation. After we have learned $\mathbf{w}$ and ξ, the classification of a new example x is done by $f(x) = argmax_{y \in \mathcal{Y}} \langle \mathbf{w}, \phi(x, y) \rangle$ with an exhaustive search of label y.

The method contains one tuning parameter which is C, the trade-off between the training error and the margin.

2.2. SVM^{struct}

In the training phase, given training examples ($\mathbf{x}^1, \mathbf{y}^1$),..., ($\mathbf{x}^n, \mathbf{y}^n$), SVM^{struct} solves the following optimization problem:

$$min_{\mathbf{w}, \xi} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^n \xi_i \tag{5}$$

$$s.t. \quad \forall (1 \leq j \leq n), \forall \mathbf{y} :$$

$$\mathbf{w}^\top \phi(\mathbf{x}^j, \mathbf{y}^j) \geq \mathbf{w}^\top \phi(\mathbf{x}^j, \mathbf{y}) + \Delta(\mathbf{y}^j, \mathbf{y}) - \xi_j,$$

where $\Delta(\mathbf{y}^j, \mathbf{y})$ is the usual loss function, calculated as the number of tag differences between $\mathbf{y}^j$ and $\mathbf{y}$. Clearly, for any feasible solution of the above optimization problem, $\sum_{i=1,\dots,n} \xi_i$ is an upper bound on the total loss. The way SVM^{struct} solves the optimization problem is described in (Tsochantaridis et al., 2005), where in each iteration the most violated constraint is added into the working set of constraints being optimized in dual formulation. The tuning parameter for this method is also C.

2.3. Maximum Margin Markov Networks

(Taskar et al., 2003) introduced the Maximum Margin Markov networks (M^3N) algorithm. In this model, a pairwise Markov network is defined as a graph $\mathcal{G} = (Y, E)$. Each edge $(i, j) \in E$ is associated with a potential function

$$\psi_{ij}(\mathbf{x}, y_i, y_j) = \exp(\sum_{k=1}^n w_k \phi_k(\mathbf{x}, y_i, y_j))$$

$$= \exp(\mathbf{w}^\top \phi(\mathbf{x}, y_i, y_j)), \tag{6}$$

where $\phi(\mathbf{x}, y_i, y_j)$ is a pairwise basis function. All edges in the graph denote the same type of interaction, so that we can define a feature map similar to the one used in SVM^{struct},

$$\phi_k(\mathbf{x}, \mathbf{y}) = \sum_{(i,j) \in E} \phi_k(\mathbf{x}, y_i, y_j). \tag{7}$$

The network encodes a joint conditional probability distribution

$$P(\mathbf{y}|\mathbf{x}) \propto \prod_{(i,j) \in E} \psi_{ij}(\mathbf{x}, y_i, y_j) = \exp(\mathbf{w}^\top \phi(\mathbf{x}, \mathbf{y})). \tag{8}$$

The weight vector $\mathbf{w}$ is selected to maximize the margin, gaining all of the advantages of the SVM framework. The primal QP for the Maximum Margin Markov network uses

the same formulation as in equation (5). However, (Taskar et al., 2003) has proposed a reparametrization of the dual variables to take advantage of the network structure of the labeling sequence problem. The dual QP is then solved using a coordinate descent method analogous to the Sequential Minimal Optimization (SMO) used for SVMs (Platt, 1999). The parameter we tuned in this model is the same C as we used in the SVM models.

Note that since both SVM^{struct} and M^{3}N solve the same QP with different techniques, we expect that both methods converge to the same optimum QP solution.

2.4. Averaged Perceptron

The perceptron algorithm for structured learning is described in (Collins, 2002). In the training phase, given training examples with a random initial weight vector **w**, the examples are iteratively processed. Let **y** be the true label sequence for input **x** and **y'** be the predicted label sequence. If a mistake is made, i.e, $\mathbf{y} \neq \mathbf{y'}$, the weight vector is updated as

$$\mathbf{w} = \mathbf{w} + \phi(\mathbf{x}, \mathbf{y}) - \phi(\mathbf{x}, \mathbf{y}'). \tag{9}$$

The final weight vector returned from the perceptron algorithm is the average of all appeared weight vectors weighted by the length of survival, i.e., number of sentences it correctly predicts before a mistake is made and the vector gets updated. The tuning parameter of the averaged perceptron algorithm is the number of iterations all sentences are processed.

2.5. SEARN

SEARN (Daumé III et al., 2006) is an algorithm for integrating search and learning to solve structured prediction problems. At training time, SEARN operates in an iterative fashion. In each iteration, it uses a known policy to create new cost-sensitive classification examples. These examples are essentially the classification decisions that a policy would need to get right in order to perform search well. These are used to learn a new classifier, i.e. a new policy. The new policy is interpolated with the old policy and the process repeats. At testing time, it uses the policy returned by the learning algorithm to construct a sequence of decisions $\widehat{\mathbf{y}}$; and a final prediction $\mathbf{y}^*$ is made by probabilistically selecting one of the sequences based on a mixing parameter β. The multiclass classifier used in the train time for SEARN is SVM^{multiclass}, and the tuning parameter in this algorithm is β.

2.6. CRF

Conditional random fields (CRFs) (Lafferty et al., 2001; Peng & McCallum, 2004) are undirected graphical models trained to maximize a conditional probability. When used for sequential labeling problems, a common special-case graph structure used is a linear chain. A linear-chain CRF with parameters **w** defines a conditional probability for a state sequence $\mathbf{y} = y_1...y_T$ given an input sequence $\mathbf{x} = x_1...x_T$ to be

$$P_{\mathbf{W}}(\mathbf{y}|\mathbf{x}) = \frac{1}{Z_{\mathbf{W}}} \exp(\mathbf{w}^\top \phi(\mathbf{x}, \mathbf{y})), \tag{10}$$

where $Z_{\mathbf{W}}$ is the normalization constant such that the sum of all the terms is one. The parameters are estimated by maximizing the following log-likelihood of the training set penalized by a Gaussian prior over the parameters,

$$\sum_i log P_{\mathbf{W}}(\mathbf{y}_i|\mathbf{x}_i) - \sum_k \frac{w_k^2}{2\sigma_k^2}, \tag{11}$$

where σ_k^2 is the variance of the Gaussian distribution and is also the tuning parameter for this method. In our experiment we used the "mallet" package (McCallum, 2002) as a CRF implementation for maximizing equation 11.

2.7. HMM

Hidden Markov Models (HMM) (Rabiner, 1989) are a traditional statistical tool for modeling generative sequences that can be characterized by an underlying process generating an observable sequence. An HMM learns a generative model over input pairs, each consisting of a sequence of observations and a sequence of labels. While enjoying much success in the past (Seymore et al., 1999; Takasu, 2003), standard HMM models have difficulty modeling multiple non-independent features. Formally, given an observation sequence we find the most probable state path for the observed sequence via the Viterbi algorithm, i.e.

$$max_{q_1 q_2 ... q_T} P(q_1 q_2 ... q_T | o_1 o_2 ... o_T). \tag{12}$$

where $Q = q_1, q_2, ..., q_T$ is the state sequence of length T, and $O = o_1, o_2, ..., o_T$ is the corresponding observations.

The transition matrix is computed as follows:

$$a_{ij} = P(q_i|q_j) = \frac{Count(q_i, q_j)}{Count(q_j)} \tag{13}$$

where $Count(q_i, q_j)$ is the number of times q_i is followed by q_j.

Second, the initial probability distribution is computed as follows:

$$\pi_i = P(q_1) = \frac{Count(q_1)}{n}, \tag{14}$$

where n is the number of training sequences.

For discrete observations such as in the case of POS tagging task, the observation matrix is computed as follows:

$$b_j(k) = P(o_k|q_j) = \frac{Count(o_k, q_j) + \alpha}{Count(q_j) + |\Sigma|\alpha} \quad (15)$$

where $Count(o_k, q_j)$ is the number of times o_k was labeled as q_j, and α is a smoothing parameter. The tuning parameter for the discrete case is α.

For cases when the observations are vectors such as OCR task, we use the Gaussian continuous-density hidden Markov model (CDHMM) to model the state emission probability,

$$b_j(k) = P(o_k|q_j) = \mathcal{N}(\mu_j, \Sigma_j) \quad (16)$$

where μ_j and Σ_j is the mean and covariance matrix of observations emitted in state q_j.

In the experiment, we used a Hidden Markov Model implementation in MATLAB by (Murphy, 1998).

3. Structured Learning Ensemble (SLE)

For an input example $\mathbf{x}$ and an ensemble of size N, predictions from different trained structured learning models, $\{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_N\}$, are combined to produce the ensemble prediction $\mathbf{y}$. The main difference between ensemble methods for the multiclass classification and the structured-output classification is the way to combine the predicted results of the base models. Since there are intrinsic structures and correlations in the output label sequence, we conjectured that the scheme of majority voting which suits well for the multi-class problem would not be sufficient. Formally, the predicted sequence of the ensemble using the majority voting scheme is computed as:

$$\mathbf{y} = \langle majority\{(\mathbf{y}_1)_1, (\mathbf{y}_2)_1, ..., (\mathbf{y}_N)_1\}, ..., majority\{(\mathbf{y}_1)_L, (\mathbf{y}_2)_L, ..., (\mathbf{y}_N)_L\}\rangle, \quad (17)$$

where L is the length of all predictions. We have proposed an alternative combination scheme, called weighted transition combination, taking into account the correlations of labels in the output sequence. First, we construct $(L-1)$ transition matrices of size $(|\Sigma| \times |\Sigma|)$, where Σ is the set of all possible labels. A transition matrix T^k gives the transition counts at the k^{th} position as follows,

$$T^k(t_i, t_j) = count_k(t_i, t_j), \forall 1 \leq k \leq (L-1), \quad (18)$$

where $count_k(t_i, t_j)$ is the number of times the label t_j is followed by t_i at the k^{th} position in the set of predicted sequences $\{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_N\}$. Similarly we define the state-weight vector that gives the count of label t_i in position k of the predicted sequences:

$$U^k(t_i) = count_k(t_i), \forall 1 \leq k \leq L \quad (19)$$

Then, the predicted sequence of SLE is given as

$$\mathbf{y} = argmax_{\mathbf{y}} \prod_{k=1}^{L-1} T^k(y_k, y_{k+1}) \prod_{k=1}^{L} U^k(y_k) \quad (20)$$

The $argmax$ in the above equation is computed using a Viterbi-like dynamic programming. In our experiment, we compare the two combination schemes, i.e. majority voting and weighted transition.

The base models that we use to construct the ensemble include models with different parameter settings from each learning method. To grow the ensemble from an initial set of models (possibly empty), we use forward stepwise selection with replacement (Caruana et al., 2004) from the set of models to find a subset of models that would yield a better performance.

Algorithm 1 SLE algorithm

Input: a set of models $\{M_1, M_2, ..., M_k\}$
a validation set V
a number of iterations S
Output: an ensemble model that yields the best performance on the validation set V

$E_0 \leftarrow Initialized\ set\ of\ models$
for i **from** 1 **to** S **do**
  $M^* \leftarrow argmax_{M_p, p \in \{1,...,k\}}$ *PERF*$(E_{i-1} \cup M', V)$
  $E_i \leftarrow E_{i-1} \cup M^*$
end for
Return $E \leftarrow argmax_{E_p, p \in \{1,...,S\}}$ *PERF*(E_p, V)

Our algorithm is described in Algorithm 1. *PERF*(E, V) is the performance of the ensemble E on the data set V such as sequence loss or token loss which will be described in the next section. In our experiments, we set the number of iterations to be $S = 200$.

We also experimented with some other variations of the ensemble method. In one variation, each base model can have their own weights to indicate its confidence level of predictions. We can either use uniformed weights for all models or use the performance on the validation set as their corresponding weights. The model weights are used as scaling factors in equation 17, 18 and 19. We also varied the initial size of the ensemble to be empty or to include the first k best base models. Finally, we also considered adding the newly constructed ensemble back to the base model selection pool. The effect of each of the above SLE design decisions is further analyzed in section 4.3.

4. Experiment Design and Analysis

In this section, we apply the algorithms described in the previous section to two well-known structured learning

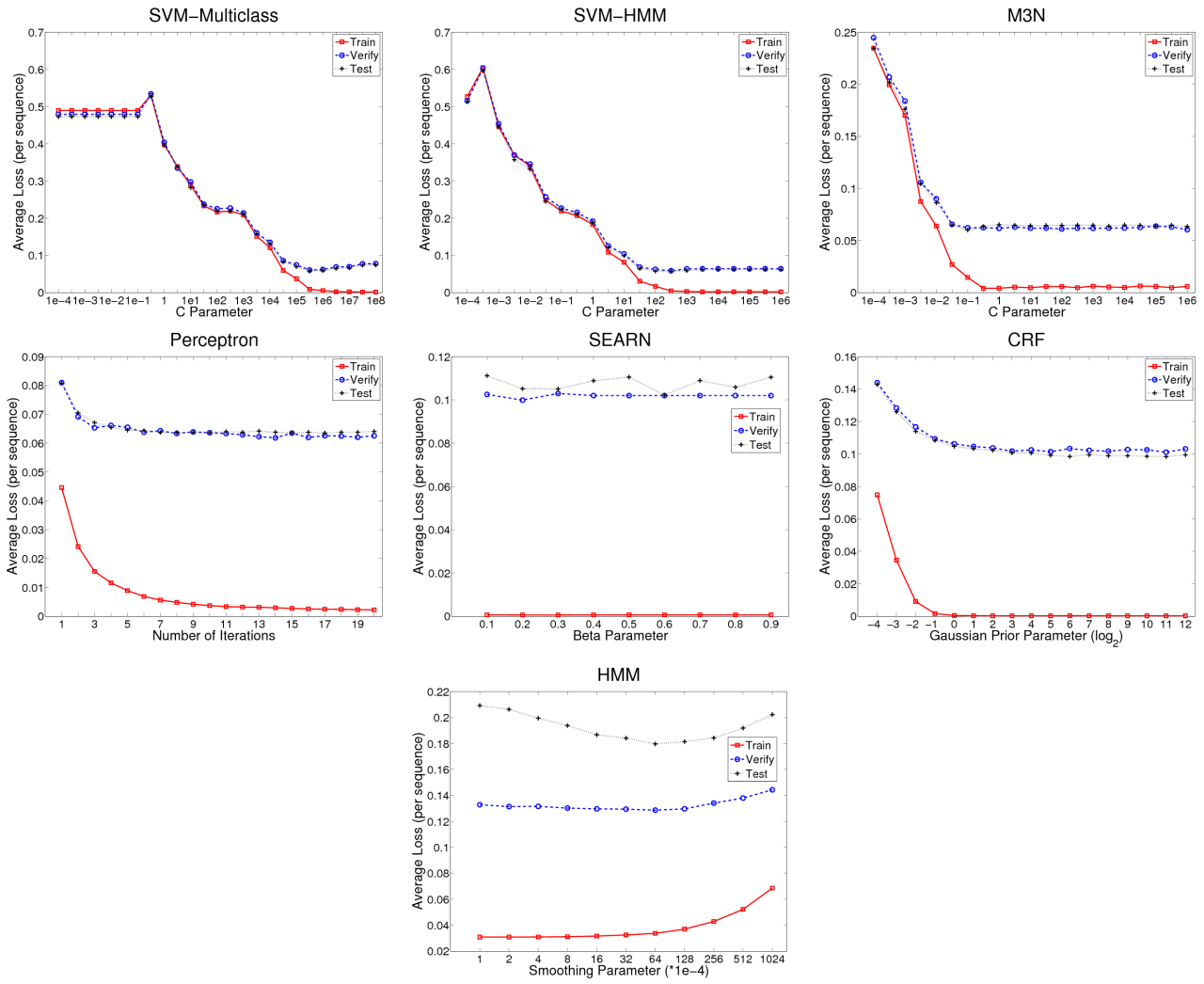

Figure 1. Various Sequence Labeling Algorithm Performances on POS Data Set with Training Set Size 2000

tasks: Part-of-speech (POS) tagging and handwritten character recognition (OCR). POS tagging consists simply of labeling each word in some text with its part of speech. Our POS data set (The Penn Treebank, 2002) is separated into five different training sizes: 500, 1000, 2000, 4000, and 8000 sentences. For each training size, we leave out 10% of the sentences as the validation data. All trained models including SLE are evaluated on a separate test set of ~ 1600 sentences. The input features for each token (i.e, a word in POS task) vary with its position in the sentence. The features are defined by the user and are usually predictors like "previous word ends with -ation". In the dataset, the total number of such simple lexical features is around 450,000.

The OCR data set contains ~ 6000 handwritten words, with average length of ~ 8 characters, from 150 human subjects, from the data set collected by Kassel (Kassel, 1995). This data set is divided into 10 folds of ~ 600 training, ~ 100 validation, and ~ 5400 testing examples. The input features for each token is a vector representation of a 16×8 binary image of a letter.

To evaluate the performance of all models, we use the average loss per sequence:

$$\text{AverageLoss} = \frac{1}{\mathrm{N}} \sum_{\mathrm{i}=1}^{\mathrm{N}} \left[\frac{1}{\mathrm{L_i}} \sum_{\mathrm{j}=1}^{\mathrm{L_i}} \mathcal{I}((\hat{\mathrm{y}}_\mathrm{i})_\mathrm{j} \neq (\mathrm{y}_\mathrm{i})_\mathrm{j}) \right], \quad (21)$$

where $\hat{\mathbf{y}}$ and $\mathbf{y}$ are the predicted sequence and the true sequence respectively; N is the total number of test examples; L_i is the length of the i^{th} sequence; and $\mathcal{I}$ is the 0-1 loss function.

Similarly, token loss (the portion of wrongly classified token) is also a valid measure of performance. Due to space

constraints we discuss our results for averaged sequence loss since it puts uniform weights on each sequence rather than individual token.

In the following sections, we present the results of various structured learning models including SLE for both tasks. The test performances of each method are reported using the model with appropriate parameter settings that gives the best result on the validation data. The range of tuning parameters of each model is chosen such that the best performance of each model is most likely observed.

4.1. The Part of Speech Tagging (POS) Task

Given the tuning parameters of individual models described in section 2, the performance of each model with respect to these parameter values is presented in Figure 1. In general, we can see the average loss of the validation set as the tuning parameter varied corresponds well with the average loss of the test set. The average loss scale (y-axis) of each plot is chosen for best visualization purposes of individual models.

The average loss of different models in percentage is presented in Table 1. For visualization purposes, Figure 2 shows the trend of decreasing average loss as the training set size increases. In Table 1, each individual model's average loss on test data is obtained by using the parameter setting with the best validation loss. We observe that among all base models studied, SVM^{struct} has the best performance on all training sets with different sizes, and our SLE method provides the overall best performance for every training set size. A t-test at the 95% confidence level indicates that the difference in performances of SLE and SVM^{struct} is significant. We also notice that SVM^{multiclass} intended as a base-line method gives competitive results on this task. One possible reason is that the input features for each word contain information of its neighboring words.

In Table 1, we notice the difference between the performances of SVM^{struct} and M^{3}N. Theoretically, the two models should converge to the same optimum since they are solving the same QP formulation with different training procedures. In our implementation of the M^{3}N algorithm, we fixed the number of iterations that the algorithm goes through the training set to 10 since the running time required by M^{3}N is much more than other algorithms. Hence, we suspect that the M^{3}N result has not converged to the optimal solution.

We also analyzed the running time of each individual model. The running time reported in this section is averaged across different parameter settings. Since we are using multiple packages or our own implementations, the absolute running time is not directly comparable. Instead,

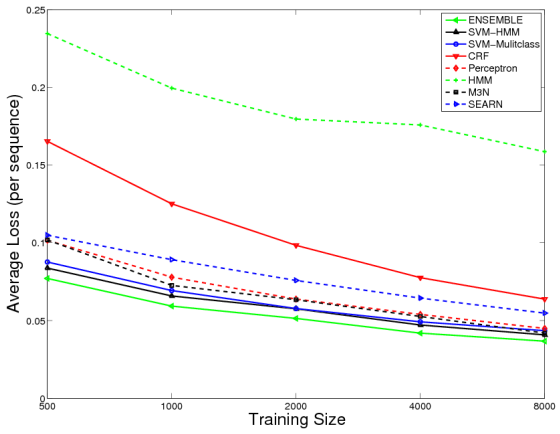

Figure 2. Model Performances on POS Data Set

Table 1. Average Loss of Algorithms on POS Data Set in Percentage

Train Size	500	1000	2000	4000	8000
SVM$^{multi.}$	8.76	6.93	5.77	4.92	4.35
SVM^{struct}	8.37	6.58	5.75	4.71	4.08
M^{3}N	10.19	7.26	6.34	5.26	4.19
Perceptron	10.16	7.79	6.38	5.39	4.49
SEARN	10.49	8.92	7.58	6.44	5.48
CRF	16.53	12.51	9.84	7.76	6.38
HMM	23.46	19.95	17.96	17.58	15.87
SLE	**7.71**	**5.93**	**5.14**	**4.19**	**3.67**

we define *time increment* in a relative sense: let the running time of a model on the smallest training set size be t_0, the time increment of that model on a training set of size s ($s \in \{500, 1000, 2000, 4000, 8000\}$) with running time t is $\frac{t}{t_0}$. Time increment represents a model's running time complexity with respect to training set size. Figure 3 graphically represents how time increment varies with the training set size for different models in log-log scale. We observe that the running time curves fall into three different groups: the first group contains only CRF whose curve is a straight line indicating that the running time is polynomial in training set size; the second group consists of SVM^{struct}, M^{3}N, and Perceptron; and the third group consists of SVM^{multiclass}, SEARN, HMM. For readers interested in the absolute running time, the real time t_0 for the models is: $SVM^{multiclass}$ 1.8m, SVM^{struct} 6.8m, M^3N 12h, Perceptron 6.4m, SEARN 2.1m, CRF 0.53h, HMM 0.23s.

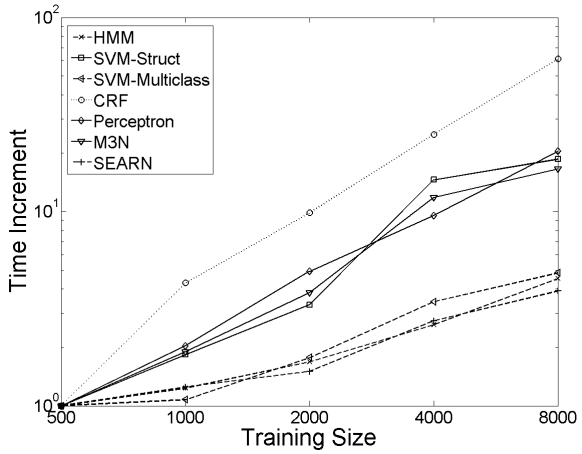

Figure 3. Various Sequence Labeling Algorithm Running Time on POS Data Set (log-log plot)

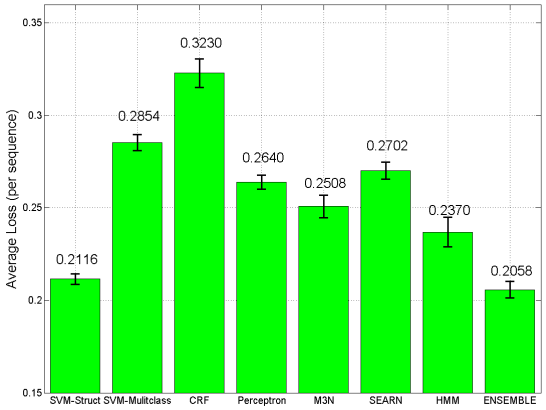

Figure 4. Model Performances on OCR Data Set

4.2. The Optical Character Recognition (OCR) Task

The relationship between model performance and parameter setting on the OCR task is similar to that of the POS task. The validation performance agrees with the test performance. Due to space constraints, we omit the figure of the average loss against parameter setting of each model.

The averaged performance of the 10-fold cross validation of all models as well as SLE is shown in Figure 4. Again, we see that SLE is the best performing sequence labeling algorithm with the average loss of 20.58%, and SVM^{struct} being the best among individual models with average loss 21.16%. Surprisingly unlike the POS task, the performance of the HMM model is reasonably good (23.70%). Hence, depending on the tasks generative model such as HMM can be as competitive as most other discriminative models. For the same reason described in section 4.1, M^{3}N performance does not converge to that of SVM^{struct}.

4.3. SLE Analysis

In Section 3, we described the various design decisions we needed to make for SLE, including 4 categories: initial ensemble size (0 or 5), base model prediction combination method (weighted transition or majority voting), model weight assignment (uniform or 1-verification loss), and whether or not we add the newly learned ensemble model back to the model selection pool. Thus, we have 16 ensemble methods in total. In order to assess the significance of each category, we present the average loss results in Table 2, where each number is the average loss of 8 models with the column category fixed. Using a t-test at the 95% confidence level, only the result difference in the combination method category is significant. As a result, we propose the weighted transition as the SLE combination mechanism.

In Table 3, we present the percentage of individual models selected in the final model learned by SLE. We observe that SVM^{struct} is not only the best performing individual model but also the most frequently selected by SLE. In addition, the frequency that a model is selected by SLE is not clearly related to its performance. Hence, all models, rather than only a few leading individual ones, contribute to the superior performance of SLE.

5. Conclusion

In this paper, we have surveyed the current state-of-art structured learning algorithms. For the first time, these methods are evaluated in uniform environment with two well-known structured learning tasks, namely the Part of Speech Tagging (POS) task and Optical Character Recognition (OCR) task. Empirical evidence suggests that SVM^{struct} provides best performance in both learning tasks. We also proposed the Structured Learning Ensemble (SLE), which combines the predictions of various base level models. Although SLE bears a resemblance of traditional ensemble method in multiclass predictions, structured learning tasks require SLE to take the correlations in output label sequences into consideration. On both POS and OCR tasks, SLE has exhibited superior performance compared with the single best model SVM^{struct}. Other ensemble techniques including bagging and boosting have been shown to work well with traditional classification problems. For future work, we will extend these techniques for structured learning problems.

Table 2. Average Loss in Percentage of Ensemble Model with Different Category Settings

POS	Init. Size		Combination		Weight		Add Model	
Train Size	0	5	wei. transition	majority	uniform	verify loss	no	yes
500	7.58	7.58	7.53	7.63	7.58	7.58	7.62	7.54
1000	5.97	5.93	5.93	5.96	5.96	5.93	5.95	5.94
2000	5.18	5.31	5.17	5.32	5.25	5.23	5.22	5.27
4000	4.31	4.28	4.26	4.33	4.31	4.27	4.28	4.30
8000	3.68	3.68	3.66	3.69	3.68	3.68	3.67	3.69
OCR	2.073	2.064	2.060	2.078	2.072	2.066	2.069	2.069

Table 3. Percentage of Individual Models Included in SLE

Data Set	SVM^{struct}	SVM$^{multi.}$	CRF	Perceptron	M^{3}N	SEARN	HMM
POS	53.55	16.53	7.79	6.59	6.54	1.81	7.19
OCR	52.27	25.70	9.47	3.61	5.97	0.79	2.20

Acknowledgments

The authors would like to thank Thorsten Joachims for his invaluable advice throughout the course of this study.

References

Caruana, R., Niculescu-Mizil, A., Crew, G., & Ksikes, A. (2004). Ensemble selection from libraries of models. *Proceedings of the 21st International Conference on Machine Learning*.

Collins, M. (2002). Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. *EMNLP*.

Crammer, K., & Singer, Y. (2001). On the algorithmic implementation of multiclass kernel-based vector machines. *Joural of Machine Learning Research, 2*, 265–292.

Daumé III, H., Langford, J., & Marcu, D. (2006). Search-based structured prediction. *Submitted to the Machine Learning Journal*.

Kassel, R. (1995). A comparison of approaches on online handwritten character recognition. Ph.D. Thesis, MIT Spoken Language System Group.

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *International Conference on Machine Learning*.

McCallum, A. K. (2002). Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu.

Murphy, K. (1998). Hidden markov model (hmm) toolbox for matlab. http://www.cs.ubc.ca/ murphyk/Software/HMM/hmm.html.

Peng, F., & McCallum, A. (2004). Accurate information extraction from research papers using conditional random fields. *HLT/NAACL*.

Platt, J. C. (1999). Using analytic qp and sparseness to speed training of support vector machines. *Proceedings of Advances in neural information processing systems* (pp. 557–563).

Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of IEEE*.

Seymore, K., McCallum, A., & Rosenfeld, R. (1999). Learning hidden Markov model structure for information extraction. *AAAI 99 Workshop on Machine Learning for Information Extraction*.

Takasu, A. (2003). Bibliographic attribute extraction from erroneous references based on a statistical model. *JCDL '03: Proceedings of the 3rd ACM/IEEE-CS joint conference on Digital libraries*.

Taskar, B., Guestrin, C., & Koller, D. (2003). Max-margin markov networks. *Advances in Neural Information Processing System 16*.

The Penn Treebank (2002). Penn's linguistic data consortium. http://www.cis.upenn.edu/ treebank.

Tsochantaridis, I., Joachims, T., Hofmann, T., & Altun, Y. (2005). Large margin methods for structured and interdependent output variables. *Joural of Machine Learning Research, 6*, 1453–1484.