Stacked Sequential Learning

William W. Cohen
Center for Automated Learning & Discovery
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
wcohen@cs.cmu.edu

Abstract

We describe a new sequential learning scheme called "stacked sequential learning". Stacked sequential learning is a meta-learning algorithm, in which an arbitrary base learning is augmented so as make it aware of the labels of nearby examples. We evaluate the method on several "sequential partitioning problems", which are characterized by long runs of identical labels. We demonstrate that on these problems, sequential stacking consistently improves the performance of non-sequential base learners; that sequential stacking often improves performance of learners (such as CRFs) that are designed specifically for sequential tasks; and that a sequentially stacked maximum-entropy learner frequently outperforms CRFs.

1 Introduction

In this paper, we will consider the application of sequential probabilistic learners to *sequential partitioning tasks*. Sequential partitioning tasks are sequential classification tasks characterized by long runs of identical labels: examples of these tasks include document analysis, video segmentation, and gene finding.

Motivated by some anomolous behavior observed for one sequential learning method on a particular partitioning task, we will derive a new learning scheme called *stacked sequential learning*. Like boosting, stacked sequential learning is a meta-learning method, in which an arbitrary base learner is augmented—in this case, by making the learner aware of the labels of nearby examples. Sequential stacking is simple to implement, can be applied to virtually any base learner, and imposes an constant overhead in training time: in our implementation, the sequentially stacked version of base learning algorithm A trains about seven times more slowly than A.

In experiments on several partitioning tasks, sequential stacking consistently improves the performance of non-sequential base learners. More surprisingly, sequential stacking also often improves performance of learners specifically designed for sequential tasks, such as conditional random fields and discriminatively trained HMMs. Finally, on our set of benchmark problems, a sequentially stacked maximum-entropy learner generally outperforms conditional random fields.

2 Motivation

2.1 A Task for Which MEMMs Fail

To motivate the novel learning method that we will describe below, we will first analyze the behavior of MEMM's on a particular real-world problem. In a recent paper [2], we evaluated a number of sequential learning methods on the problem of recognizing the "signature" section of an email message. Each line of an email message was represented with a set of hand-crafted features, such as "line contains a possible phone number", "line is blank", etc. Each email message in the dataset was represented as a vector $\mathbf{x}$ of feature-vectors $x_1, \ldots, x_\ell$, where x_i is the feature-vector representation of the i-th line of the message. A line was labeled as *positive* if it was part of a signature section, and *negative* otherwise. The labels for a message were represented as another vector $\mathbf{y}$, where y_i is the label for line i.

The dataset contains about 33,013 labeled lines from 617 email messages. About 10% of the lines are labeled "positive". Signature sections always fall at the end of a message, usually in the last 10 lines. In the experiments below, the data was split into a training set (of 438 sequences/emails), and a test set with the remaining sequences, and we used the "basic" feature set from Carvalho & Cohen (i.e., without "window fea-

tures").

The complete dataset is represented as a set S of examples $S = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_t, \mathbf{y}_t), \ldots, (\mathbf{x}_m, \mathbf{y}_m)\}$. *Sequential learning* is the problem of learning, from such a dataset, a *sequential classifier*—i.e., a function f such that $f(\mathbf{x})$ produces a vector of class labels $\mathbf{y}$. Clearly, any ordinary non-sequential learning algorithm can be used for sequential learning, by ignoring the sequential nature of the data[1].

In the previous paper [2], we reported results for several non-sequential and sequential learners on the signature-detection problem, including a non-sequential maximum entropy learner [1] (henceforth ME) and conditional random fields [8] (henceforth CRFs). Another plausible sequential learning method to apply to this task are *maximum-entropy Markov models* (MEMMs) [9], also called *maximum-entropy taggers* [11] and recurrent sliding windows [4]. In this model, the conditional probability of a label sequence $\mathbf{y}$ given an instance sequence $\mathbf{x}$ is defined to be

$$\Pr(\mathbf{y}|\mathbf{x}) = \prod_i \Pr(y_i|y_{i-1}, x_i) \quad (1)$$

The *local model* $\Pr(y_i|y_{i-1}, x_i)$ is learned as follows. First one constructs an *extended dataset*, which is a collection of non-sequential examples of the form $((x_i, y_{i-1}), y_i)$, where (x_i, y_{i-1}) denotes an instance in which the original feature vector for x_i is augmented by adding a feature for y_{i-1}. We will call (x_i, y_{i-1}) is an *extended instance*, and call y_{i-1} a *history feature*. Note that y_i is the class label for the extended instance.

After constructing extended instances, one trains a maximum-entropy conditional model from the extended dataset. Inference is done with a Viterbi search to find the best label sequence $\mathbf{y}$ according to Equation 1.

MEMMs have a number of nice properties. Relative the more recently-proposed CRF model, MEMMs are easy to implement, and (since no inference is done at learning time) relatively quick to train. MEMMs can also be easily generalized by replacing the local model with one that uses a longer "history" of k previous labels—i.e., a model of the form $\Pr(y_i|y_{i-1}, \ldots, y_{i-k}, x_i)$—and replacing the Viterbi search with a beam search. Such a learner scales well with the history size and number of possible classes y.

[1]Specifically, one could build a dataset of non-sequential examples $(x_{t,i}, y_{t,i})$ from S, and use it to train a classifier g that maps a feature-vector x to a label y. One can then use g to classify each instance x_i in the vector $\mathbf{x} = \langle x_1, \ldots, x_n$ separately, ignoring its sequential position, and append the resulting predictions y_i into an output vector $\mathbf{y}$.

Method	Noise	Error	(Min Error)
ME		3.47	(3.20)
MEMM		31.83	(4.26)
CRF		1.17	(1.17)
MEMM	10%	2.18	(2.18)
CRF	10%	1.85	(1.84)

Table 1: Performance of several sequential learners on the signature-detection problem.

Unfortunately, as Table 1 shows, MEMMs perform extremely badly on the signature-detection problem described above, with an error rate many times the error rate of CRFs. In fact, on this problem, MEMMs perform much worse than the non-sequential maximum-entropy learner ME. (We used the implementations of ME, MEMMs, and CRFs provided by Minorthird [10], which uses Gaussian priors and a limited-memory quasi-Newton method for optimization. A limit of 50 optimization iterations was also used, although this limit does not substantially change the result of this section.)

The MEMM's performance is somewhat better if one is allowed to change the threshold used to classify examples. Letting $f(x_i)$ be probability that $y_i = +$ in the Viterbi classification of $\mathbf{x}$, we computed, for each learner, the threshold θ such the rule $[(y_i = +) \Leftrightarrow (f(x_i) > \theta)]$ gives the lowest test error rate. The column labeled "Min Error" in Table 1 gives this result. (Of course, since the computation of θ was done using the test data, this is only a lower bound on attainable error rate.) The "Min Error" for MEMMs is much lower than the error for MEMMs with the default threshold, but still higher than either non-sequential ME or CRFs.

2.2 Analysis

The literature suggests several possible explanations for these results. For instance, Lafferty *et al* [8] show that MEMMs can represent only a proper subset of the distributions that can be represented by CRFs (the "label bias problem"). However, "label bias" does not explain why MEMMs perform worse than non-sequential ME, since MEMMs clearly can represent a proper superset of the distributions that ME can represent. Klein and Manning [7] describe an "observation bias problem", in which MEMMs give too little weight to the "history" features $y_{i-1}, \ldots, y_{i-k}$. Error analysis on the signature-detection task suggests that the opposite is happening here: relative to the weights assigned by a CRF, MEMM is actually giving *too much* weight to the history features, and too little to the fea-

Stacked Sequential Learning.

Parameters: a history size W_h, a future size W_f, and a cross-validation parameter K.

Learning algorithm: Given a sample $S = \{(\mathbf{x}_t, \mathbf{y}_t)\}$, and a sequential learning algorithm A:

1. Construct a sample of predictions $\hat{\mathbf{y}}_t$ for each $\mathbf{x}_t \in S$ as follows:
   (a) Split S into K equal-sized disjoint subsets $S_1, \ldots, S_K$
   (b) For $j = 1, \ldots, K$, let $f_j = A(S - S_j)$
   (c) Let $\hat{S} = \{(\mathbf{x}_t, \hat{\mathbf{y}}_t) : \hat{\mathbf{y}} = f_j(\mathbf{x}_t) \text{ and } \mathbf{x_t} \in S_j\}$
2. Construct an extended dataset S' of instances $(\mathbf{x'}_t, \mathbf{y}_t)$ by converting each $\mathbf{x}_t$ to $\mathbf{x'}_t$ as follows: $\mathbf{x_t}' = \langle x'_1, \ldots, x'_{\ell_t} \rangle$ where $x'_i = (x_i, \hat{y}_{i-W_h}, \ldots, \hat{y}_{i+W_f})$ and $\hat{y}_i$ is the i-th component of $\hat{\mathbf{y}}_t$, the label vector paired with $\mathbf{x}_t$ in $\hat{S}$.
3. Return two functions: $f = A(S)$ and $f' = A(S')$.

Inference algorithm: given an instance vector $\mathbf{x}$:

1. Let $\hat{\mathbf{y}} = f(\mathbf{x})$
2. Carry out Step 2 above to produce an extended instance $\mathbf{x}'$ (using $\hat{\mathbf{y}}$ in place of $\hat{\mathbf{y}}_t$).
3. Return $f'(\mathbf{x}')$.

Table 2: The sequential stacking meta-learning algorithm.

tures from x_i. The conjecture that the history features are being overweighted is also consistent with the empirical observation that on many test email messages, the learned MEMM makes a false positive classification somewhere before the signature starts, and then "gets stuck" and marks every subsequent line as part of a signature.

To test this theory, we encouraged the MEMM to downweight the history features by adding noise to the training (not test) data, as follows. For each training email/sequence $\mathbf{x}$, we consider each feature-vector $x_i \in \mathbf{x}$ in turn, and toss a coin with a 10% chance of landing "heads". If the coin flip comes up "heads", we swap x_i with some other feature-vector x_j chosen uniformly from $\mathbf{x}$. Adding this "sequence noise" almost doubles the error rate for CRFs, but greatly reduces the error rate for MEMMs. (Of course, this type of noise does not affect non-sequential ME.) This experiment further supports the hypothesis that MEMM is overweighting history features.

3 Stacked Sequential Learning

3.1 Description

The poor results for MEMM described above can be intuitively explained as a mismatch between the data used to *train* the local models of the MEMM, and the data used to *test* the model. With noise-free training data, it is *always* the case that a signature line is followed by more signature lines, so it is not especially surprising that the MEMM's local model tends to weight this feature heavily. However, this regularity need not always hold for the test data, which is drawn from *predictions* made by the local model on different examples.

In theory, of course, this training/test mismatch is compensated for by the Viterbi search, which is in turn driven by the confidence estimates produced by the local model. However, if the assumptions of the theory are violated (for instance, if there are high-order interactions not accounted for by the maximum-entropy model), the local model's confidence estimates may be incorrect.

To correct the training/test mismatch, it is sufficient to modify the the extended dataset so that the true previous class y_{i-1} in an extended instance (x_i, y_{i-1}) is replaced by a *predicted* previous class $\hat{y}_{i-1}$. Below we will outline one way to do this.

Assume that one is given a sample $S = \{(\mathbf{x}_t, \mathbf{y}_t)\}$ of size m, and a sequential learning algorithm A. Previous work on a meta-learning method called *stacking* [12] suggests the following scheme for constructing a sample of $(\mathbf{x}, \hat{\mathbf{y}})$ pairs so that $\hat{\mathbf{y}}$ is similar to the prediction produced by an f learned by A on a size-m sample that does not include $\mathbf{x}$. First, partition S into K equal-sized disjoint subsets $S_1, \ldots, S_K$, and learn K functions $f_1, \ldots, f_K$, where $f_j = A(S - S_j)$. Then, construct the set

$$\hat{S} = \{(\mathbf{x}_t, \hat{\mathbf{y}}_t) : \hat{\mathbf{y}} = f_j(\mathbf{x}_t) \text{ and } \mathbf{x_t} \in S_j\}$$

In other words, $\hat{S}$ pairs each $\mathbf{x}_t$ with the $\hat{\mathbf{y}}_t$ associated with performing a K-fold cross-validation on S.

This suggests the learning algorithm of Table 2. This method begins with a sample S and a sequential learning method A. In the discussion below we will assume that A is ME, used for sequential data.

Using S, A, and cross-validation techniques, one first pairs with each $\mathbf{x}_t \in S$ the vector $\hat{\mathbf{y}}_t$ associated with performing cross-validation with ME. These predictions are then used to create a dataset S' of extended instances $\mathbf{x}'$, which in the simplest case, are simply vectors composed of instances of the form $(x_i, \hat{y}_{i-1})$, where $\hat{y}_{i-1}$ is the $(i-1)$-th label in $\hat{\mathbf{y}}$.

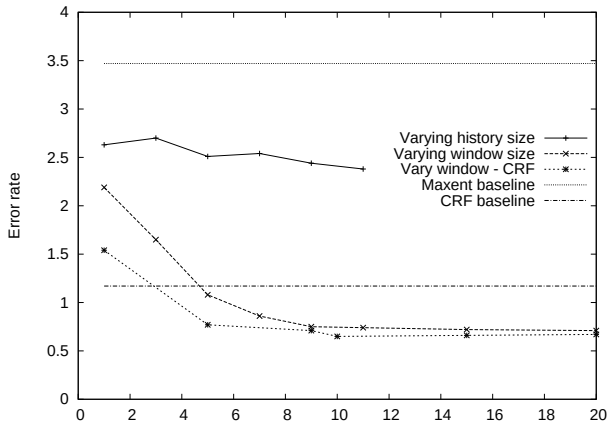


Figure 1: Stacked sequential learning, varying history size W_h and window size ($W = W_h = W_f$). The base learning algorithm A is maximum-entropy (ME), unless otherwise stated.

The extended examples S' are then used to train a model $f' = A(S')$. If A is the non-sequential maximum-entropy learner, this step is similar to the process of building a "local model" for an MEMM: the difference is that the history features added to x_i are derived not from the true history of x_i, but approximate the off-sample predictions of an ME classifier.

At inference time, f' must be run on examples that have been extended by adding prediction features $\hat{y}$. To keep the "test" distribution similar to the "training" distribution, f will not be used as the inner loop of a Viterbi or beam-search process; instead, the predictions $\hat{y}$ are produced using a non-sequential maximum-entropy model f that is learned from S. The algorithm of Table 2 simply generalizes this idea from ME to an arbitrary sequential learner.

In our experiments, we introduced some small but important refinements to this process. First, each "history feature" $\hat{y}$ added to an extended example is not simply a predicted class, but a numeric value indicating the log-odds of that class. This makes accessible to f' the confidences previously used by the Viterbi search. Second, the extended instance constructed from x_i includes $\hat{y}_i$ as well as $\hat{y}_{i-1}$, by default.

3.2 Initial results

Stacked sequential learning with ME as the base learner (henceforth s-ME) and $K = 5$ obtains an error rate of 2.63% on the signature-detection task. This is less than the baseline ME method (3.20%) but still higher than CRFs (1.17%). However, three extensions to s-ME are straightforward to implement, and dramatically improve performance.

More past labels. Like MEMMs, s-ME can efficiently handle a large "history" of previous predicted classes. (In fact, s-ME can handle large histories without resorting to beam search, since the only impact of more history features is to add new features to the extended instances.) On the signature-detection task, increasing the history size reduces error to 2.38% (with a history size of 11) as is shown in Figure 1.

Past and future labels. Unlike MEMMs, the extended instance for x_i can include predicted classes not only of previous instances, but also of "future" instances—instances that follow x_i in the sequence $\mathbf{x}$. We explored different "window sizes" for s-ME, where a "window size" of W means that the W previous and W following predicted labels are added to each extended instance. This reduces error rates substantially, to only 0.71%. This is a 46% reduction from CRF's error rate of 1.17%. The improvement is also statistically significant.[2]

Different base learners. Stacked sequential learning can be applied to any learner; in particular, since the extended examples are sequential, it can be applied any sequential learner. We evaluated stacked sequential CRFs (s-CRFs) with varying window sizes on this problem. As shown in Figure 1, s-CRFs also outperform CRFs, and again, the difference is both substantial and statistically significant. However, with large window sizes, there is little difference in performance between s-CRF and s-ME.

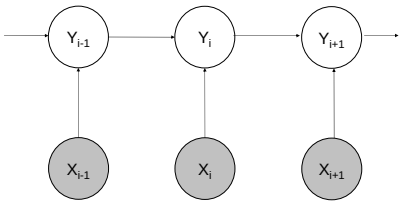
3.3 Discussion

A graphical view of a MEMMs is shown in Part(a) of Figure 2. We use the usual convention in which nodes for known values are shaded. Each node is associated with a maximum-entropy conditional model which defines a probability distribution given its input values.

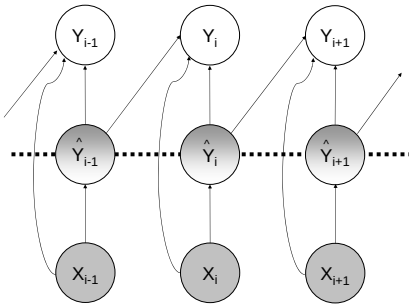
Part (b) of the figure presents a similar graphical view of the classifier learned by sequential stacking. Inference in this model is done in two stages: first the middle layer is inferred from the bottom later, then the top layer is inferred from the middle layer. The nodes in the middle layer are partly shaded to indicate that their hybrid status—they are considered outputs by the model f, and inputs by the model f'.

One way to interpret the hybrid layer is as a means of making the inference more robust. If the middle-layer nodes were treated as ordinary unobserved variables, the top-layer conditional model (f') would rely heavily on the confidence assessments of the lower-layer model

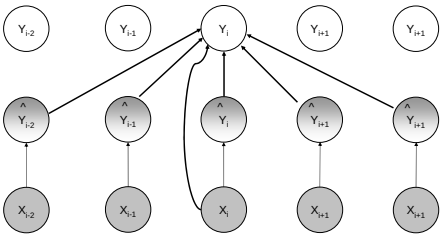
[2]Specifically, a two-tailed paired t-test rejects with $>$ 95% confidence the null hypothesis that the difference in error rate between s-ME and CRF on a randomly selected sequence $\mathbf{x}$ has a mean of zero.



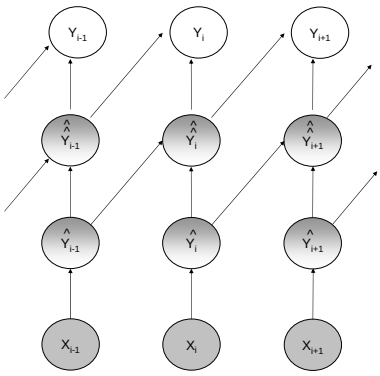
(a) Maximum-entropy Markov model (MEMM)



(b) Sequential stacking



(c) Sequential stacking with width $W = 2$



(d) Two-level sequential stacking

Figure 2: Graphical views of alternative sequential-stacking schemes.

(f). Forcing f' treat these variables as *observed* quantities allows f' to develop its own model of how the $\hat{y}$ predictions made by f correlate with the actual outputs y. This allows f' to accept or downweight f's predictions, as appropriate.

Part (c) of the figure shows a sequential stacking model with a window of $W_h = W_f = 2$. To simplify the figure, only the edges that eventually lead to the node Y_i are shown.

Part (d) of the figure shows another plausible extension of sequential stacking, in which each $\hat{y}$ is replaced with a better approximation of y—namely, the output of sequential stacking itself. This "deeper" stacking scheme can be implemented quite easily, for instance by applying the sequential stacking scheme to the base learner s-ME. However, our initial experiments were discouraging: for instance, the depth-two learner s-(s-ME) has a slightly higher error rate than s-ME (3.04%). The limited amount of training data available for the lowest-level models may be an issue: with $K = 5$, for instance, only 64% of the total data is available on the lowest-level cross-validation runs.

To conclude our discussion, we note that as described, sequential stacking increases run-time of the base learning method by a approximately a constant factor of $K+2$. (To see this, note sequential stacking requires training $K+2$ classifiers: the classifiers $f_1, \ldots, f_k$ used in cross-validation, and the final classifiers f and f'.) When data is plentiful but training time is limited, it is also possible to simply split the original dataset S into two disjoint halves S_1 and S_2, and train two classifiers f and f' from S_1 and S_2' respectively (where S_2' is S_2, extended with the predictions produced by f). This scheme leaves training time approximately unchanged for a linear-time base learner, and decreases training time for any base learner that requires superlinear time.

Task	MEMM	ME	CRF	s-ME	s-CRF
A/aigen	53.61	8.02	20.35	6.91	**5.78**
A/ainn	70.09	6.61	2.14	3.65	**1.67**
A/aix	13.86	5.02	6.83	**4.59**	11.79
T/aigen	0.30	2.60	2.39	1.92	**0.00**
T/ainn	1.36	1.39	0.28	**0.00**	0.28
T/aix	3.51	1.25	5.26	**0.05**	4.44
1/video	**11.39**	12.66	12.66	12.66	13.92
2/video	8.86	8.86	7.59	**3.80**	7.59
mailsig	31.83	3.47	1.17	1.08	**0.77**

Table 3: Comparision of different sequential algorithms on a set of nine benchmark tasks.

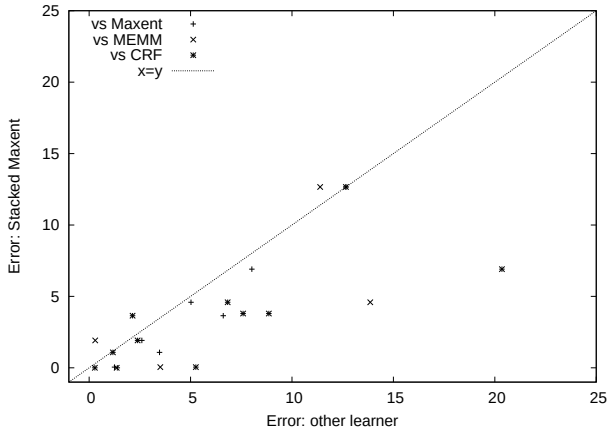


Figure 3: Comparision of the error rates for s-ME with the error rates of ME, MEMM, and CRFs.

4 Experimental Results

4.1 Additional Problems

We also evaluated non-sequential ME, MEMMs, CRFs, s-ME, and s-CRFs on several other sequential partitioning tasks. For stacking, we used $K = 5$ and a window size of $W_h = W_f = 5$ on all problems. These were the only parameter values explored in this section, and no changes were made to the sequential stacking algorithm, which was developed based on observations made from the signature-detection task only.

One set of tasks involved classifying lines from FAQ documents with labels like "header", "question", "answer", and "trailer". We used the same three tasks (ai-general, ai-neural-nets, and aix) considered by Dietterich *et al* [5] and the features adopted by McCallum *et al* [9]. The data consists of 5-7 long sequences, each corresponding to a single FAQ document; in total, each task contains between 8,965 aand 12,757 labeled lines. Our current implementation of sequential stacking only supports binary labels, so we considered the two labels "trailer" (T) and "answer" (A) as separate tasks for each FAQ, leading to a total of six new benchmarks.

Another set of tasks were video segmentation task, in which the goal is to take a sequence of video "shots" (a sequence of adjacent frames taken from one camera) and classify them into categories such as "anchor", "news" and "weather". This dataset contains 12 sequences, each corresponding to a single video clip. There are a total of 418 shots, and about 700 features, which are produced by applying LDA to a 5x5, 125-bin RGB color histogram of the central frame of the shot. (This data was provided by Yik-Cheung Tam and Ming-yu Chen.) We constructed two separate video partitioning tasks, corresponding to the two most common labels.

All eight of these additional tasks are similar to the signature-detection task in that they contain long runs of identical labels, leading to strong regularities in constructed history features. Error rates for the learning methods on these eight tasks, in addition to the previous signature-detection task, are shown in Table 3. In each case a single train/test split was used to evaluate error rates. The bold-faced entries are the lowest error rate on a row.

We observe that MEMM suffers extremely high error rates on two of the new tasks (finding "answer" lines for ai-general and ai-neural-nets), suggesting that the "anomolous" behavior shown in signature-detection may not be uncommon, at least in sequential partitioning tasks.

Also, comparing s-ME to ME, we see that s-ME improves the error rate in 8 of 9 tasks, and leaves it unchanged once. Furthermore, s-ME has a lower error rate than CRFs 7 of 9 times. Overall, s-ME seems to be at preferable to either of three older approaches (ME, MEMMs, and CRFS). This is made somewhat more apparent by the scatter plot of Figure 3. On this plot, each point is places so the y-axis position is the error of s-ME, and the x-axis position is the error of an earlier learner; thus points below the line $y = x$ are cases where s-ME outperforms another learner. (For readability, the range of the x axis is truncated—it does not include the highest error rates of MEMM.)

Stacking also improves CRF on some problems, but the effect is not as consistent: s-CRF improves the error rate on 5 of 9 tasks, leaves it unchanged twice, and increases the error rate twice. In the table, one of the two stacked learners has the lowest error rate on 8 of the 9 tasks.

Task	VP	VPHMM	s-VP	s-VPHMM
A/aigen	7.87	12.09	**7.33**	12.09
A/ainn	6.59	10.26	**3.76**	10.26
A/aix	5.50	4.86	**3.61**	5.95
J/aigen	1.68	2.16	**0.18**	2.16
J/ainn	1.44	1.93	**0.19**	1.93
J/aix	3.40	1.95	1.16	**1.01**
1/video	**11.39**	13.92	**11.39**	**11.39**
2/video	6.33	25.32	**3.80**	6.33
mailsig	3.40	1.95	1.16	**1.01**

Table 4: Comparision of different sequential algorithms on a set of nine benchmark tasks.

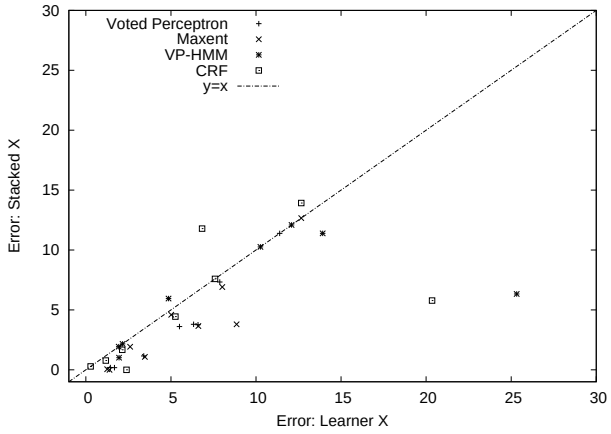


Figure 4: Comparision of the error rates various algorithms with and without sequential stacking.

4.2 Additional Base Learners

We conducted the same experiments with two margin-based base learners: the non-sequential voted perceptron algorithm (VP) [6] and a voted-perceptron based training scheme for HMMs proposed by Collins (VP-HMMs) [3]. Table 4 shows the results for these methods, and their sequentially-stacked versions. Both the sequential and non-sequential voted perceptrons were run for 20 epochs.

In this case, s-VP outperforms both VP and VPHMM on all nine problems. The s-VPHMM has lower error rate than the VPHMM 4 times, a higher error rate once, and the same error rate 4 times.

An overview of the improvements obtained by sequential stacking on these problems is shown in Figure 4 and Table 5. The scatter plot shows the error rate of ME plotted against the error rate of ME, the error rate of s-VP plotted against VP, and similarly for s-VPHMM vs VPHMM and CRFS vs s-CRFs. The plot shows a plausible pattern: sequential stacking nearly always improves the performance on the non-sequential learners (ME and VP) but improves performance of the sequential learners (CRFs and VPHMMs) less consistently. This pattern is confirmed by a series of sign tests performed on pairs of learners, which are summarized in Table 5.

Method A	W-L-T s-A *vs.* A	null hypothesis	confidence rejected
ME	8-0-1	E[err(s-A)-err(A)]$\geq$0	>0.98
VP	8-0-1	E[err(s-A)-err(A)]$\geq$0	>0.98
VPHMM	4-4-1	E[err(s-A)-err(A)]>0	>0.98
CRF	5-2-2	E[err(s-A)-err(A)]>0	>0.92

Table 5: Comparison of stacked *vs* unstacked learners, using a one-tailed sign test on error rates obtained on the nine benchmark problems.

5 Conclusions

Sequential partitioning tasks are sequential classification tasks characterized by long runs of identical labels: examples of these tasks include document analysis, video segmentation, and gene finding. In this paper, we have evaluated the performance of certain well-studied sequential probabilistic learners to sequential partitioning tasks. It was observed that MEMMs sometimes obtain extremely high error rates. Error analysis suggests that this problem is neither due to "label bias" [8] nor "observation bias" [7], but to a mismatch between the data used to train the MEMM's local model, and the data on which the MEMM's local model is tested. In particular, since MEMMs are trained on "true" labels and tested on "predicted" labels, the strong correlations between adjacent labels associated sequential partitioning tasks can be misleading to the MEMM's learning method.

Motivated by these issues, we derived a novel method in which cross-validation is used correct this mismatch. The end result is a meta-learning scheme called *stacked sequential learning*. Sequential stacking is simple to implement, can be applied to virtually any base learner, and imposes an constant overhead in learning time (the constant being the number of cross-validation folds plus two). In experiments on several partitioning tasks, sequential stacking consistently improves the performance of two non-sequential base learners, often dramatically. On our set of benchmark problems, sequential stacking with a maximum-entropy learner as the base learner outperforms CRFs 7 of 9 times. Perhaps more surprisingly, sequential stacking also often improves performance of learners specifically designed for sequential tasks, such as conditional random fields and discriminately trained HMMs.

Some initial experiments on a named entity recognition problem suggest that sequential stacking does not improve performance on non-partitioning problems; however, in future work, we plan to explore this issue with more detailed experimentation. We also plan to extend our implementation to handle non-binary label sets.

Acknowledgements

The author wishes to thank many friends and colleagues for input, in particular Vitor Carvalho and David McAllister. He is also grateful to Yik-Cheung Tam and Ming-yu Chen for providing the video-segmentation data. This work was supported by Information Processing Technology Office (IPTO) of the Defense Advanced Research Projects Agency (DARPA).

References

[1] A. L. Berger, V. J. D. Pietra, and S. A. D. Pietra. A maximum entropy approach to natural language processing. *Comput. Linguist.*, 22(1):39–71, 1996.

[2] V. Carvalho and W. W. Cohen. Learning to extract signature and reply lines from email. In *Proceedings of the Conference on Email and Anti-Spam 2004*, Mountain View, California, 2004.

[3] M. Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2002.

[4] T. G. Dietterich. Machine learning for sequential data: A review. In *Structural, Syntactic, and Statistical Pattern Recognition*, pages 15–30. Springer Verlag, New York, 2002.

[5] T. G. Dietterich, A. Ashenfelter, and Y. Bulatov. Training conditional random fields via gradient tree boosting. In *Proceedings of the 21th International Conference on Machine Learning (ICML)*, 2004.

[6] Y. Freund and R. E. Schapire. Large margin classification using the perceptron algorithm. In *Computational Learing Theory*, pages 209–217, 1998.

[7] D. Klein and C. D. Manning. Conditional structure versus conditional estimation in nlp models. In *Workshop on Empirical Methods in Natural Language Processing (EMNLP)*, 2002.

[8] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning (ICML-2001)*, Williams, MA, 2001.

[9] A. McCallum, D. Freitag, and F. Pereira. Maximum entropy markov models for information extraction and segmentation. In *Proceedings of the International Conference on Machine Learning (ICML-2000)*, pages 591–598, Palo Alto, CA, 2000.

[10] Minorthird: Methods for identifying names and ontological relations in text using heuristics for inducing regularities from data. http://minorthird.sourceforge.net, 2004.

[11] A. Ratnaparkhi. Learning to parse natural language with maximum entropy models. *Machine Learning*, 34, 1999.

[12] D. H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.