A High-Performance Semi-Supervised Learning Method for Text Chunking

Rie Kubota Ando† **Tong Zhang**‡

IBM T.J. Watson Research Center
Yorktown Heights, NY 10598, U.S.A.

†rie1@us.ibm.com ‡tongz@us.ibm.com

Abstract

In machine learning, whether one can build a more accurate classifier by using unlabeled data (*semi-supervised learning*) is an important issue. Although a number of semi-supervised methods have been proposed, their effectiveness on NLP tasks is not always clear. This paper presents a novel semi-supervised method that employs a learning paradigm which we call *structural learning*. The idea is to find "what good classifiers are like" by learning from thousands of automatically generated auxiliary classification problems on unlabeled data. By doing so, the common predictive structure shared by the multiple classification problems can be discovered, which can then be used to improve performance on the target problem. The method produces performance higher than the previous best results on CoNLL'00 syntactic chunking and CoNLL'03 named entity chunking (English and German).

1 Introduction

In supervised learning applications, one can often find a large amount of unlabeled data without difficulty, while labeled data are costly to obtain. Therefore, a natural question is whether we can use unlabeled data to build a more accurate classifier, given the same amount of labeled data. This problem is often referred to as *semi-supervised learning*.

Although a number of semi-supervised methods have been proposed, their effectiveness on NLP tasks is not always clear. For example, *co-training* (Blum and Mitchell, 1998) automatically bootstraps labels, and such labels are not necessarily reliable (Pierce and Cardie, 2001). A related idea is to use *Expectation Maximization* (EM) to *impute* labels. Although useful under some circumstances, when a relatively large amount of labeled data is available, the procedure often degrades performance (e.g. Merialdo (1994)). A number of bootstrapping methods have been proposed for NLP tasks (e.g. Yarowsky (1995), Collins and Singer (1999), Riloff and Jones (1999)). But these typically assume a very small amount of labeled data and have not been shown to improve state-of-the-art performance when a large amount of labeled data is available.

Our goal has been to develop a general learning framework for reliably using unlabeled data to improve performance irrespective of the amount of labeled data available. It is exactly this important and difficult problem that we tackle here.

This paper presents a novel semi-supervised method that employs a learning framework called *structural learning* (Ando and Zhang, 2004), which seeks to discover shared *predictive structures* (i.e. what good classifiers for the task are like) through jointly learning multiple classification problems on unlabeled data. That is, we systematically create thousands of problems (called *auxiliary problems*) relevant to the target task using unlabeled data, and train classifiers from the automatically generated 'training data'. We learn the commonality (or structure) of such many classifiers relevant to the task, and use it to improve performance on the target task. One example of such auxiliary problems for *chunking* tasks is to 'mask' a word and predict whether it is "people" or not from the context, like language modeling. Another example is to predict the prediction of some classifier trained for the target task. These auxiliary classifiers can be adequately learned since we have very large amounts of 'training data' for them, which we automatically generate from a very large amount of unlabeled data.

The contributions of this paper are two-fold. First, we present a novel robust semi-supervised method based on a new learning model and its application to chunking tasks. Second, we report higher performance than the previous best results on syntactic chunking (the CoNLL'00 corpus) and named entity chunking (the CoNLL'03 English and German corpora). In particular, our results are obtained by using unlabeled data as the *only* additional resource while many of the top systems rely on hand-crafted resources such as large name gazetteers or even rule-based post-processing.

2 A Model for Learning Structures

This work uses a linear formulation of structural learning. We first briefly review a standard linear prediction model and then extend it for structural learning. We sketch an optimization algorithm using SVD and compare it to related methods.

2.1 Standard linear prediction model

In the standard formulation of supervised learning, we seek a *predictor* that maps an input vector $\mathbf{x} \in \mathcal{X}$ to the corresponding output $y \in \mathcal{Y}$. *Linear prediction models* are based on real-valued predictors of the form $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$, where $\mathbf{w}$ is called a *weight vector*. For binary problems, the sign of the linear prediction gives the class label. For k-way classification (with $k > 2$), a typical method is *winner takes all*, where we train one predictor per class and choose the class with the highest output value.

A frequently used method for finding an accurate predictor $\hat{f}$ is regularized *empirical risk minimization (ERM)*, which minimizes an empirical loss of the predictor (with regularization) on the n training examples $\{(\mathbf{X}_i, Y_i)\}$:

$$\hat{f} = \arg\min_f \left(\sum_{i=1}^n L(f(\mathbf{X}_i), Y_i) + r(f) \right) .$$

$L(\cdot)$ is a *loss function* to quantify the difference between the prediction $f(\mathbf{X}_i)$ and the true output Y_i, and $r(\cdot)$ is a regularization term to control the model complexity. ERM-based methods for discriminative learning are known to be effective for NLP tasks such as chunking (e.g. Kudoh and Matsumoto (2001), Zhang and Johnson (2003)).

2.2 Linear model for structural learning

We present a linear prediction model for structural learning, which extends the traditional model to multiple problems. Specifically, we assume that there exists a *low-dimensional predictive structure* shared by multiple prediction problems. We seek to discover this structure through *joint empirical risk minimization* over the multiple problems.

Consider m problems indexed by $\ell \in \{1, \ldots, m\}$, each with n_ℓ samples $(\mathbf{X}_i^\ell, Y_i^\ell)$ indexed by $i \in \{1, \ldots, n_\ell\}$. In our joint linear model, a predictor for problem ℓ takes the following form

$$f_\ell(\Theta, \mathbf{x}) = \mathbf{w}_\ell^T \mathbf{x} + \mathbf{v}_\ell^T \Theta \mathbf{x} \ , \ \ \Theta\Theta^T = \mathbf{I} \ , \qquad (1)$$

where we use $\mathbf{I}$ to denote the identity matrix. Matrix Θ (whose rows are orthonormal) is the common *structure parameter* shared by all the problems; $\mathbf{w}_\ell$ and $\mathbf{v}_\ell$ are weight vectors specific to each prediction problem ℓ. The idea of this model is to discover a common low-dimensional predictive structure (shared by the m problems) parameterized by the projection matrix Θ. In this setting, the goal of structural learning may also be regarded as *learning a good feature map* $\Theta\mathbf{x}$ — a low-dimensional feature vector parameterized by Θ.

In joint ERM, we seek Θ (and weight vectors) that minimizes the empirical risk summed over all the problems:

$$[\hat{\Theta}, \{\hat{f}_\ell\}] = \arg\min_{\Theta, \{f_\ell\}} \sum_{\ell=1}^m \left(\sum_{i=1}^{n_\ell} \frac{L(f_\ell(\Theta, \mathbf{X}_i^\ell), Y_i^\ell)}{n_\ell} + r(f_\ell) \right) . \qquad (2)$$

It can be shown that using joint ERM, we can reliably estimate the optimal joint parameter Θ as long as m is large (even when each n_ℓ is small). This is the key reason why structural learning is effective. A formal PAC-style analysis can be found in (Ando and Zhang, 2004).

2.3 Alternating structure optimization (ASO)

The optimization problem (2) has a simple solution using SVD when we choose square regularization:

$r(f_\ell) = \lambda \|\mathbf{w}_\ell\|_2^2$, where the regularization parameter λ is given. For clarity, let $\mathbf{u}_\ell$ be a weight vector for problem ℓ such that: $\mathbf{u}_\ell = \mathbf{w}_\ell + \Theta^T \mathbf{v}_\ell$. Then, (2) becomes the minimization of the joint empirical risk written as:

$$\sum_{\ell=1}^{m} \left(\sum_{i=1}^{n_\ell} \frac{L(\mathbf{u}_\ell^T \mathbf{X}_i^\ell, Y_i^\ell)}{n_\ell} + \lambda \|\mathbf{u}_\ell - \Theta^T \mathbf{v}_\ell\|_2^2 \right) . \quad (3)$$

This minimization can be approximately solved by the following alternating optimization procedure:

- Fix $(\Theta, \{\mathbf{v}_\ell\})$, and find m predictors $\{\mathbf{u}_\ell\}$ that minimizes the joint empirical risk (3).
- Fix m predictors $\{\mathbf{u}_\ell\}$, and find $(\Theta, \{\mathbf{v}_\ell\})$ that minimizes the joint empirical risk (3).
- Iterate until a convergence criterion is met.

In the first step, we train m predictors independently. It is the second step that couples all the problems. Its solution is given by the SVD (singular value decomposition) of the predictor matrix $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_m]$: the rows of the optimum Θ are given by the most significant *left singular vectors*[1] of $\mathbf{U}$. Intuitively, the optimum Θ captures the maximal commonality of the m predictors (each derived from $\mathbf{u}_\ell$). These m predictors are updated using the new structure matrix Θ in the next iteration, and the process repeats.

Figure 1 summarizes the algorithm sketched above, which we call the *alternating structure optimization (ASO)* algorithm. The formal derivation can be found in (Ando and Zhang, 2004).

2.4 Comparison with existing techniques

It is important to note that this SVD-based ASO (SVD-ASO) procedure is fundamentally different from the usual principle component analysis (PCA), which can be regarded as dimension reduction in the *data space* $\mathcal{X}$. By contrast, the dimension reduction performed in the SVD-ASO algorithm is on the *predictor space* (a set of predictors). This is possible because we observe multiple predictors from multiple learning tasks. If we regard the observed predictors as sample points of the predictor distribution in the predictor space (corrupted with estimation error, or noise), then SVD-ASO can be interpreted as finding the "principle components" (or commonality) of these predictors (i.e., "what good predictors are like"). Consequently the method *directly* looks for low-dimensional structures with the highest predictive power. By contrast, the principle components of input data in the data space (which PCA seeks) may *not* necessarily have the highest predictive power.

```
Input: training data $\{(\mathbf{X}_i^\ell, Y_i^\ell)\}$ $(\ell = 1, \ldots, m)$
Parameters: dimension $h$ and regularization param $\lambda$
Output: matrix $\Theta$ with $h$ rows
Initialize: $\mathbf{u}_\ell = 0$ $(\ell = 1 \ldots m)$, and arbitrary $\Theta$
iterate
  for $\ell = 1$ to $m$ do
    With fixed $\Theta$ and $\mathbf{v}_\ell = \Theta \mathbf{u}_\ell$, solve for $\hat{\mathbf{w}}_\ell$:
      $\hat{\mathbf{w}}_\ell = \arg\min_{\mathbf{w}_\ell} \Big[ \sum_{i=1}^{n_\ell} \frac{L(\mathbf{w}_\ell^T \mathbf{X}_i^\ell + (\mathbf{v}_\ell^T \Theta) \mathbf{X}_i^\ell, Y_i^\ell)}{n_\ell}$
                  $+ \lambda \|\mathbf{w}_\ell\|_2^2 \Big]$
    Let $\mathbf{u}_\ell = \hat{\mathbf{w}}_\ell + \Theta^T \mathbf{v}_\ell$
  endfor
  Compute the SVD of $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_m]$.
  Let the rows of $\Theta$ be the $h$ left singular vectors of $\mathbf{U}$
    corresponding to the $h$ largest singular values.
until converge
```

Figure 1: SVD-based Alternating Structure Optimization (SVD-ASO) Algorithm

The above argument also applies to the feature generation from unlabeled data using LSI (e.g. Ando (2004)). Similarly, Miller et al. (2004) used word-cluster memberships induced from an unannotated corpus as features for named entity chunking. Our work is related but more general, because we can explore additional information from unlabeled data using many different auxiliary problems. Since Miller et al. (2004)'s experiments used a proprietary corpus, direct performance comparison is not possible. However, our preliminary implementation of the word clustering approach did not provide any improvement on our tasks. As we will see, our starting performance is already high. Therefore the additional information discovered by SVD-ASO appears crucial to achieve appreciable improvements.

3 Semi-supervised Learning Method

For semi-supervised learning, the idea is to *create* many auxiliary prediction problems (relevant to the task) from unlabeled data so that we can learn the

[1] In other words, Θ is computed so that the best low-rank approximation of $\mathbf{U}$ in the least square sense is obtained by projecting $\mathbf{U}$ onto the row space of Θ; see e.g. Golub and Loan (1996) for SVD.

shared structure Θ (useful for the task) using the ASO algorithm. In particular, we want to create auxiliary problems with the following properties:

- *Automatic labeling*: we need to automatically generate various "labeled" data for the auxiliary problems from unlabeled data.
- *Relevancy*: auxiliary problems should be related to the target problem. That is, they should share a certain predictive structure.

The final classifier for the target task is in the form of (1), a linear predictor for structural learning. We fix Θ (learned from unlabeled data through auxiliary problems) and optimize weight vectors $\mathbf{w}$ and $\mathbf{v}$ on the given labeled data. We summarize this semi-supervised learning procedure below.

1. Create training data $\widetilde{Z}_\ell = \{(\widetilde{\mathbf{X}}_j, \widetilde{Y}_j^\ell)\}$ for each auxiliary problem ℓ from unlabeled data $\{\widetilde{\mathbf{X}}_j\}$.
2. Compute Θ from $\{\widetilde{Z}_\ell\}$ through SVD-ASO.
3. Minimize the empirical risk on the labeled data: $\hat{f} = \arg\min_f \sum_{i=1}^n \frac{L(f(\Theta,\mathbf{X}_i),Y_i)}{n} + \lambda\|\mathbf{w}\|_2^2$, where $f(\Theta,\mathbf{x}) = \mathbf{w}^T\mathbf{x} + \mathbf{v}^T\Theta\mathbf{x}$ as in (1).

3.1 Auxiliary problem creation

The idea is to discover useful features (which do not necessarily appear in the labeled data) from the unlabeled data through learning auxiliary problems. Clearly, auxiliary problems more closely related to the target problem will be more beneficial. However, even if some problems are less relevant, they will not degrade performance severely since they merely result in some irrelevant features (originated from irrelevant Θ-components), which ERM learners can cope with. On the other hand, potential gains from relevant auxiliary problems can be significant. In this sense, our method is robust.

We present two general strategies for generating useful auxiliary problems: one in a completely unsupervised fashion, and the other in a partially-supervised fashion.

3.1.1 Unsupervised strategy

In the first strategy, we regard some observable substructures of the input data $\mathcal{X}$ as auxiliary class labels, and try to predict these labels using other parts of the input data.

Ex 3.1 Predict words. *Create auxiliary problems by regarding the word at each position as an auxiliary label, which we want to predict from the context. For instance, predict whether a word is "Smith" or not from its context. This problem is relevant to, for instance, named entity chunking since knowing a word is "Smith" helps to predict whether it is part of a name. One binary classification problem can be created for each possible word value (e.g., "IBM", "he", "get", $\cdots$). Hence, many auxiliary problems can be obtained using this idea.*

More generally, given a feature representation of the input data, we may mask some features as unobserved, and learn classifiers to predict these 'masked' features based on other features that are not masked. The automatic-labeling requirement is satisfied since the auxiliary labels are observable to us. To create relevant problems, we should choose to (mask and) predict features that have good correlation to the target classes, such as words on text tagging/chunking tasks.

3.1.2 Partially-supervised strategy

The second strategy is motivated by co-training. We use two (or more) distinct feature maps: $\mathbf{\Phi}_1$ and $\mathbf{\Phi}_2$. First, we train a classifier F_1 for the target task, using the feature map $\mathbf{\Phi}_1$ and the labeled data. The auxiliary tasks are to predict the behavior of this classifier F_1 (such as predicted labels) on the unlabeled data, by using the other feature map $\mathbf{\Phi}_2$. Note that unlike co-training, we only use the classifier as a means of creating auxiliary problems that meet the relevancy requirement, instead of using it to bootstrap labels.

Ex 3.2 Predict the top-k choices of the classifier. *Predict the combination of k (a few) classes to which F_1 assigns the highest output (confidence) values. For instance, predict whether F_1 assigns the highest confidence values to* CLASS1 *and* CLASS2 *in this order. By setting $k = 1$, the auxiliary task is simply to predict the label prediction of classifier F_1. By setting $k > 1$, fine-grained distinctions (related to intrinsic sub-classes of target classes) can be learned. From a c-way classification problem, $c!/(c-k)!$ binary prediction problems can be created.*

4 Algorithms Used in Experiments

Using auxiliary problems introduced above, we study the performance of our semi-supervised learning method on named entity chunking and syntactic chunking. This section describes the algorithmic aspects of the experimental framework. The task-specific setup is described in Sections 5 and 6.

4.1 Extension of the basic SVD-ASO algorithm

In our experiments, we use an extension of SVD-ASO. In NLP applications, features have natural grouping according to their types/origins such as 'current words', 'parts-of-speech on the right', and so forth. It is desirable to perform a localized optimization for each of such natural feature groups. Hence, we associate each feature group with a submatrix of structure matrix Θ. The optimization algorithm for this extension is essentially the same as SVD-ASO in Figure 1, but with the SVD step performed separately for each group. See (Ando and Zhang, 2004) for the precise formulation. In addition, we regularize only those components of $\mathbf{w}_\ell$ which correspond to the non-negative part of $\mathbf{u}_\ell$. The motivation is that positive weights are usually directly related to the target concept, while negative ones often yield much less specific information representing 'the others'. The resulting extension, in effect, only uses the positive components of $\mathbf{U}$ in the SVD computation.

4.2 Chunking algorithm, loss function, training algorithm, and parameter settings

As is commonly done, we encode chunk information into word tags to cast the chunking problem to that of sequential word tagging. We perform Viterbi-style decoding to choose the word tag sequence that maximizes the sum of tagging confidence values.

In all settings (including baseline methods), the loss function is a modification of the Huber's robust loss for regression: $L(p, y) = \max(0, 1 - py)^2$ if $py \geq -1$; and $-4py$ otherwise; with square regularization ($\lambda = 10^{-4}$). One may select other loss functions such as SVM or logistic regression. The specific choice is not important for the purpose of this paper. The training algorithm is *stochastic gradient descent*, which is argued to perform well for regularized convex ERM learning formulations (Zhang, 2004).

As we will show in Section 7.3, our formulation is relatively insensitive to the change in h (row-dimension of the structure matrix). We fix h (for each feature group) to 50, and use it in all settings.

The most time-consuming process is the training of m auxiliary predictors on the unlabeled data (computing $\mathbf{U}$ in Figure 1). Fixing the number of iterations to a constant, it runs in linear to m and the number of unlabeled instances and takes hours in our settings that use more than 20 million unlabeled instances.

4.3 Baseline algorithms

Supervised classifier For comparison, we train a classifier using the same features and algorithm, but without unlabeled data ($\Theta = \mathbf{0}$ in effect).

Co-training We test co-training since our idea of partially-supervised auxiliary problems is motivated by co-training. Our implementation follows the original work (Blum and Mitchell, 1998). The two (or more) classifiers (with distinct feature maps) are trained with labeled data. We maintain a pool of q unlabeled instances by random selection. The classifier proposes labels for the instances in this pool. We choose s instances for each classifier with high confidence while preserving the class distribution observed in the initial labeled data, and add them to the labeled data. The process is then repeated. We explore q=50K, 100K, s=50,100,500,1K, and commonly-used feature splits: 'current vs. context' and 'current+left-context vs. current+right-context'.

Self-training Single-view bootstrapping is sometimes called *self-training*. We test the basic self-training[2], which replaces multiple classifiers in the co-training procedure with a single classifier that employs all the features.

co/self-training oracle performance To avoid the issue of parameter selection for the co- and self-training, we report their best possible *oracle performance*, which is the best F-measure number among all the co- and self-training parameter settings including the choice of the number of iterations.

[2] We also tested "self-training with bagging", which Ng and Cardie (2003) used for co-reference resolution. We omit results since it did not produce better performance than the supervised baseline.

- words, parts-of-speech (POS), character types, 4 characters at the beginning/ending in a 5-word window.
- words in a 3-syntactic chunk window.
- labels assigned to two words on the left.
- bi-grams of the current word and the label on the left.
- labels assigned to previous occurrences of the current word.

Figure 2: Feature types for named entity chunking. POS and syntactic chunk information is provided by the organizer.

5 Named Entity Chunking Experiments

We report named entity chunking performance on the CoNLL'03 shared-task[3] corpora (English and German). We choose this task because the original intention of this shared task was to test the effectiveness of semi-supervised learning methods. However, it turned out that none of the top performing systems used unlabeled data. The likely reason is that the number of labeled data is relatively large (>200K), making it hard to benefit from unlabeled data. We show that our ASO-based semi-supervised learning method (hereafter, *ASO-semi*) can produce results appreciably better than all of the top systems, by using unlabeled data as the *only* additional resource. In particular, we do not use any gazetteer information, which was used in all other systems.

The CoNLL corpora are annotated with four types of named entities: persons, organizations, locations, and miscellaneous names (e.g., “World Cup”). We use the official training/development/test splits. Our unlabeled data sets consist of 27 million words (English) and 35 million words (German), respectively. They were chosen from the same sources – Reuters and ECI Multilingual Text Corpus – as the provided corpora but disjoint from them.

5.1 Features

Our feature representation is a slight modification of a simpler configuration (without any gazetteer) in (Zhang and Johnson, 2003), as shown in Figure 2. We use POS and syntactic chunk information provided by the organizer.

5.2 Auxiliary problems

As shown in Figure 3, we experiment with auxiliary problems from Ex 3.1 and 3.2: “Predict current (or previous or next) words”; and “Predict *top-2* choices of the classifier” using feature splits ‘left context vs. the others’ and ‘right context vs. the others’. For word-prediction problems, we only consider the instances whose current words are either nouns or adjectives since named entities mostly consist of these types. Also, we leave out all but at most 1000 binary prediction problems of each type that have the largest numbers of positive examples to ensure that auxiliary predictors can be adequately learned with a sufficiently large number of examples. The results we report are obtained by using all the problems in Figure 3 unless otherwise specified.

# of aux. problems	Auxiliary labels	Features used for learning aux problems
1000	previous words	all but previous words
1000	current words	all but current words
1000	next words	all but next words
72	F_1's top-2 choices	$\mathbf{\Phi}_2$ (all but left context)
72	F_2's top-2 choices	$\mathbf{\Phi}_1$ (left context)
72	F_3's top-2 choices	$\mathbf{\Phi}_4$ (all but right context)
72	F_4's top-2 choices	$\mathbf{\Phi}_3$ (right context)

Figure 3: Auxiliary problems used for named entity chunking. 3000 problems ‘mask’ words and predict them from the other features on unlabeled data. 288 problems predict classifier F_i's predictions on unlabeled data, where F_i is trained with labeled data using feature map $\mathbf{\Phi}_i$. There are 72 possible top-2 choices from 9 classes (beginning/inside of four types of name chunks and ‘outside’).

5.3 Named entity chunking results

methods	test data	F	diff. from supervised prec.	recall	F
English, small (10K examples) training set					
ASO-semi	dev.	**81.25**	*+10.02*	*+7.00*	*+8.51*
co/self oracle		73.10	+0.32	+0.39	+0.36
ASO-semi	test	**78.42**	*+9.39*	*+10.73*	*+10.10*
co/self oracle		69.63	+0.60	+1.95	+1.31
English, all (204K) training examples					
ASO-semi	dev.	**93.15**	*+2.25*	*+3.00*	*+2.62*
co/self oracle		90.64	+0.04	+0.20	+0.11
ASO-semi	test	**89.31**	*+3.20*	*+4.51*	*+3.86*
co/self oracle		85.40	−0.04	−0.05	−0.05
German, all (207K) training examples					
ASO-semi	dev.	**74.06**	*+7.04*	*+10.19*	*+9.22*
co/self oracle		66.47	−2.59	+4.39	+1.63
ASO-semi	test	**75.27**	*+4.64*	*+6.59*	*+5.88*
co/self oracle		70.45	−1.26	+2.59	+1.06

Figure 4: Named entity chunking results. No gazetteer. F-measure and performance improvements over the supervised baseline in precision, recall, and F. For co- and self-training (baseline), the *oracle* performance is shown.

Figure 4 shows results in comparison with the supervised baseline in six configurations, each trained

[3] http://cnts.uia.ac.be/conll2003/ner

with one of three sets of labeled training examples: a small English set (10K examples randomly chosen), the entire English training set (204K), and the entire German set (207K), tested on either the development set or test set. ASO-semi significantly improves both precision and recall in all the six configurations, resulting in improved F-measures over the supervised baseline by +2.62% to +10.10%.

Co- and self-training, at their *oracle performance*, improve recall but often degrade precision; consequently, their F-measure improvements are relatively low: -0.05% to +1.63%.

Comparison with top systems As shown in Figure 5, ASO-semi achieves higher performance than the top systems on both English and German data. Most of the top systems boost performance by external hand-crafted resources such as: large gazetteers[4]; a large amount (2 million words) of *labeled* data manually annotated with finer-grained named entities (FIJZ03); and rule-based post processing (KSNM03). Hence, we feel that our results, obtained by using unlabeled data as the only additional resource, are encouraging.

System	Eng.	Ger.	Additional resources
ASO-semi	**89.31**	**75.27**	*unlabeled data*
FIJZ03	88.76	72.41	gazetteers; 2M-word labeled data (English)
CN03	88.31	65.67	gazetteers (English); (also very elaborated features)
KSNM03	86.31	71.90	rule-based post processing

Figure 5: Named entity chunking. F-measure on the test sets. Previous best results: FIJZ03 (Florian et al., 2003), CN03 (Chieu and Ng, 2003), KSNM03 (Klein et al., 2003).

6 Syntactic Chunking Experiments

Next, we report syntactic chunking performance on the CoNLL'00 shared-task[5] corpus. The training and test data sets consist of the Wall Street Journal corpus (WSJ) sections 15–18 (212K words) and section 20, respectively. They are annotated with eleven types of syntactic chunks such as noun phrases. We use the WSJ articles in 1991 (15 million words) from the TREC corpus as the unlabeled data.

· uni- and bi-grams of words and POS in a 5-token window.
· word-POS bi-grams in a 3-token window.
· POS tri-grams on the left and right.
· labels of the two words on the left and their bi-grams.
· bi-grams of the current word and two labels on the left.

Figure 6: Feature types for syntactic chunking. POS information is provided by the organizer.

	prec.	recall	$F_{\beta=1}$
supervised	93.83	93.37	93.60
ASO-semi	**94.57**	**94.20**	**94.39 (+0.79)**
co/self oracle	93.76	93.56	93.66 (+0.06)

Figure 7: Syntactic chunking results.

6.1 Features and auxiliary problems

Our feature representation is a slight modification of a simpler configuration (without linguistic features) in (Zhang et al., 2002), as shown in Figure 6. We use the POS information provided by the organizer. The types of auxiliary problems are the same as in the named entity experiments. For word predictions, we exclude instances of punctuation symbols.

6.2 Syntactic chunking results

As shown in Figure 7, ASO-semi improves both precision and recall over the supervised baseline. It achieves 94.39% in F-measure, which outperforms the supervised baseline by 0.79%. Co- and self-training again slightly improve recall but slightly degrade precision at their oracle performance, which demonstrates that it is not easy to benefit from unlabeled data on this task.

Comparison with the previous best systems As shown in Figure 8, ASO-semi achieves performance higher than the previous best systems. Though the space constraint precludes providing the detail, we note that ASO-semi outperforms all of the previous top systems in both precision and recall. Unlike named entity chunking, the use of external resources on this task is rare. An exception is the use of output from a grammar-based full parser as features in ZDJ02+, which our system does not use. KM01 and CM03 boost performance by classifier combinations. SP03 trains conditional random fields for NP

[4] Whether or not gazetteers are useful depends on their coverage. A number of top-performing systems used their own gazetteers in addition to the organizer's gazetteers and reported significant performance improvements (e.g., FIJZ03, CN03, and ZJ03).

[5] http://cnts.uia.ac.be/conll2000/chunking

	all	NP	description
ASO-semi	**94.39**	**94.70**	*+unlabeled data*
KM01	93.91	94.39	SVM combination
CM03	93.74	94.41	perceptron in two layers
SP03	–	94.38	conditional random fields
ZDJ02	93.57	93.89	generalized Winnow
ZDJ02+	94.17	94.38	+full parser output

Figure 8: Syntactic chunking F-measure. Comparison with previous best results: KM01 (Kudoh and Matsumoto, 2001), CM03 (Carreras and Marquez, 2003), SP03 (Sha and Pereira, 2003), ZDJ02 (Zhang et al., 2002).

(noun phrases) only. ASO-semi produces higher NP chunking performance than the others.

7 Empirical Analysis

7.1 Effectiveness of auxiliary problems

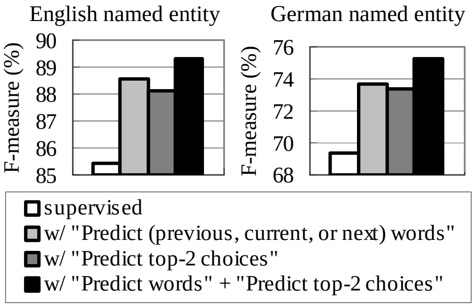

Figure 9: Named entity F-measure produced by using individual types of auxiliary problems. Trained with the entire training sets and tested on the test sets.

Figure 9 shows F-measure obtained by computing Θ from individual types of auxiliary problems on named entity chunking. Both types – "Predict words" and "Predict top-2 choices of the classifier" – are useful, producing significant performance improvements over the supervised baseline. The best performance is achieved when Θ is produced from all of the auxiliary problems.

7.2 Interpretation of Θ

To gain insights into the information obtained from unlabeled data, we examine the Θ entries associated with the feature 'current words', computed for the English named entity task. Figure 10 shows the features associated with the entries of Θ with the largest values, computed from the 2000 unsupervised auxiliary problems: "Predict previous words" and "Predict next words". For clarity, the figure only shows words beginning with upper-case letters (i.e., likely to be names in English). Our method captures the spirit of predictive word-clustering but is more general and effective on our tasks.

row#	Features corresponding to significant Θ entries	Interpretation
4	Ltd, Inc, Plc, International, Ltd., Association, Group, Inc.	organizations
7	Co, Corp, Co., Company, Authority, Corp., Services	organizations
9	PCT, N/A, Nil, Dec, BLN, Avg, Year-on-year, UNCH	no names
11	New, France, European, San, North, Japan, Asian, India	locations
15	Peter, Sir, Charles, Jose, Paul, Lee, Alan, Dan, John, James	persons
26	June, May, July, Jan, March, August, September, April	months

Figure 10: Interpretation of Θ computed from word-prediction (unsupervised) problems for named entity chunking.

It is possible to develop a general theory to show that the auxiliary problems we use are helpful under reasonable conditions. The intuition is as follows. Suppose we split the features into two parts $\mathbf{\Phi}_1$ and $\mathbf{\Phi}_2$ and predict $\mathbf{\Phi}_1$ based on $\mathbf{\Phi}_2$. Suppose features in $\mathbf{\Phi}_1$ are correlated to the class labels (but not necessarily correlated among themselves). Then, the auxiliary prediction problems are related to the target task, and thus can reveal useful structures of $\mathbf{\Phi}_2$. Under some conditions, it can be shown that features in $\mathbf{\Phi}_2$ with similar predictive performance tend to map to similar low-dimensional vectors through Θ. This effect can be empirically observed in Figure 10 and will be formally shown elsewhere.

7.3 Effect of the Θ dimension

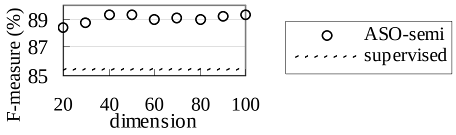

Figure 11: F-measure in relation to the row-dimension of Θ. English named entity chunking, test set.

Recall that throughout the experiments, we fix the row-dimension of Θ (for each feature group) to 50. Figure 11 plots F-measure in relation to the row-dimension of Θ, which shows that the method is relatively insensitive to the change of this parameter, at least in the range which we consider.

8 Conclusion

We presented a novel semi-supervised learning method that learns the most predictive low-dimensional feature projection from unlabeled data using the structural learning algorithm SVD-ASO. On CoNLL'00 syntactic chunking and CoNLL'03 named entity chunking (English and German), the method exceeds the previous best systems (including those which rely on hand-crafted resources) by using unlabeled data as the only additional resource.

The key idea is to create auxiliary problems automatically from unlabeled data so that predictive structures can be learned from that data. In practice, it is desirable to create as many auxiliary problems as possible, as long as there is some reason to believe in their relevancy to the task. This is because the risk is relatively minor while the potential gain from relevant problems is large. Moreover, the auxiliary problems used in our experiments are merely possible examples. One advantage of our approach is that one may design a variety of auxiliary problems to learn various aspects of the target problem from unlabeled data. Structural learning provides a framework for carrying out possible new ideas.

Acknowledgments

Part of the work was supported by ARDA under the NIMD program PNWD-SW-6059.

References

Rie Kubota Ando and Tong Zhang. 2004. A framework for learning predictive structures from multiple tasks and unlabeled data. Technical report, IBM. RC23462.

Rie Kubota Ando. 2004. Semantic lexicon construction: Learning from unlabeled data via spectral analysis. In *Proceedings of CoNLL-2004*.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *proceedings of COLT-98*.

Xavier Carreras and Lluis Marquez. 2003. Phrase recognition by filtering and ranking with perceptrons. In *Proceedings of RANLP-2003*.

Hai Leong Chieu and Hwee Tou Ng. 2003. Named entity recognition with a maximum entropy approach. In *Proceedings CoNLL-2003*, pages 160–163.

Michael Collins and Yoram Singer. 1999. Unsupervised models for named entity classification. In *Proceedings of EMNLP/VLC'99*.

Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. 2003. Named entity recognition through classifier combination. In *Proceedings CoNLL-2003*, pages 168–171.

Gene H. Golub and Charles F. Van Loan. 1996. Matrix computations third edition.

Dan Klein, Joseph Smarr, Huy Nguyen, and Christopher D. Manning. 2003. Named entity recognition with character-level models. In *Proceedings CoNLL-2003*, pages 188–191.

Taku Kudoh and Yuji Matsumoto. 2001. Chunking with support vector machines. In *Proceedings of NAACL 2001*.

Bernard Merialdo. 1994. Tagging English text with a probabilistic model. *Computational Linguistics*, 20(2):155–171.

Scott Miller, Jethran Guinness, and Alex Zamanian. 2004. Name tagging with word clusters and discriminative training. In *Proceedings of HLT-NAACL-2004*.

Vincent Ng and Claire Cardie. 2003. Weakly supervised natural language learning without redundant views. In *Proceedings of HLT-NAACL-2003*.

David Pierce and Claire Cardie. 2001. Limitations of co-training for natural language learning from large datasets. In *Proceedings of EMNLP-2001*.

Ellen Riloff and Rosie Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of AAAI-99*.

Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of HLT-NAACL'03*.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of ACL-95*.

Tong Zhang and David E. Johnson. 2003. A robust risk minimization based named entity recognition system. In *Proceedings CoNLL-2003*, pages 204–207.

Tong Zhang, Fred Damerau, and David E. Johnson. 2002. Text chunking based on a generalization of Winnow. *Journal of Machine Learning Research*, 2:615–637.

Tong Zhang. 2004. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *ICML 04*, pages 919–926.