Learning Predictive Structures for Semantic Role Labeling of NomBank

Chang Liu and **Hwee Tou Ng**
Department of Computer Science
National University of Singapore
3 Science Drive 2, Singapore 117543
{liuchan1, nght}@comp.nus.edu.sg

Abstract

This paper presents a novel application of Alternating Structure Optimization (ASO) to the task of Semantic Role Labeling (SRL) of noun predicates in NomBank. ASO is a recently proposed linear multi-task learning algorithm, which extracts the common structures of multiple tasks to improve accuracy, via the use of auxiliary problems. In this paper, we explore a number of different auxiliary problems, and we are able to significantly improve the accuracy of the NomBank SRL task using this approach. To our knowledge, our proposed approach achieves the highest accuracy published to date on the English NomBank SRL task.

1 Introduction

The task of Semantic Role Labeling (SRL) is to identify predicate-argument relationships in natural language texts in a domain-independent fashion. In recent years, the availability of large human-labeled corpora such as PropBank (Palmer et al., 2005) and FrameNet (Baker et al., 1998) has made possible a statistical approach of identifying and classifying the arguments of verbs in natural language texts. A large number of SRL systems have been evaluated and compared on the standard data set in the CoNLL shared tasks (Carreras and Marquez, 2004; Carreras and Marquez, 2005), and many systems have performed reasonably well. Compared to the previous CoNLL shared tasks (noun phrase bracketing, chunking, clause identification, and named entity recognition), SRL represents a significant step towards processing the semantic content of natural language texts.

Although verbs are probably the most obvious predicates in a sentence, many nouns are also capable of having complex argument structures, often with much more flexibility than its verb counterpart. For example, compare *affect* and *effect*:

> [$_{subj}$ Auto prices] [$_{arg-ext}$ greatly] [$_{pred}$ affect] [$_{obj}$ the PPI].
>
> [$_{subj}$ Auto prices] have a [$_{arg-ext}$ big] [$_{pred}$ effect] [$_{obj}$ on the PPI].
>
> The [$_{pred}$ effect] [$_{subj}$ of auto prices] [$_{obj}$ on the PPI] is [$_{arg-ext}$ big].
>
> [$_{subj}$ The auto prices'] [$_{pred}$ effect] [$_{obj}$ on the PPI] is [$_{arg-ext}$ big].

The arguments of noun predicates can often be more easily omitted compared to the verb predicates:

> The [$_{pred}$ effect] [$_{subj}$ of auto prices] is [$_{arg-ext}$ big].
>
> The [$_{pred}$ effect] [$_{obj}$ on the PPI] is [$_{arg-ext}$ big].
>
> The [$_{pred}$ effect] is [$_{arg-ext}$ big].

With the recent release of NomBank (Meyers et al., 2004), it becomes possible to apply machine learning techniques to the task. So far we are aware of only one English NomBank-based SRL system (Jiang and Ng, 2006), which uses the maximum entropy classifier, although similar efforts are reported on the Chinese NomBank by (Xue, 2006)

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 208–215,
Prague, Czech Republic, June 2007.

and on FrameNet by (Pradhan et al., 2004) using a small set of hand-selected nominalizations. Noun predicates also appear in FrameNet semantic role labeling (Gildea and Jurafsky, 2002), and many FrameNet SRL systems are evaluated in Senseval-3 (Litkowski, 2004).

Semantic role labeling of NomBank is a multi-class classification problem by nature. Using the *one-vs-all* arrangement, that is, one binary classifier for each possible outcome, the SRL task can be treated as multiple binary classification problems. In the latter view, we are presented with the opportunity to exploit the *common structures* of these related problems. This is known as *multi-task learning* in the machine learning literature (Caruana, 1997; Ben-David and Schuller, 2003; Evgeniou and Pontil, 2004; Micchelli and Pontil, 2005; Maurer, 2006).

In this paper, we apply *Alternating Structure Optimization* (ASO) (Ando and Zhang, 2005a) to the semantic role labeling task on NomBank. ASO is a recently proposed linear multi-task learning algorithm based on empirical risk minimization. The method requires the use of multiple auxiliary problems, and its effectiveness may vary depending on the specific auxiliary problems used. ASO has been shown to be effective on the following natural language processing tasks: text categorization, named entity recognition, part-of-speech tagging, and word sense disambiguation (Ando and Zhang, 2005a; Ando and Zhang, 2005b; Ando, 2006).

This paper makes two significant contributions. First, we present a novel application of ASO to the SRL task on NomBank. We explore the effect of different auxiliary problems, and show that learning predictive structures with ASO results in significantly improved SRL accuracy. Second, we achieve accuracy higher than that reported in (Jiang and Ng, 2006) and advance the state of the art in SRL research.

The rest of this paper is organized as follows. We give an overview of NomBank and ASO in Sections 2 and 3 respectively. The baseline linear classifier is described in detail in Section 4, followed by the description of the ASO classifier in Section 5, where we focus on exploring different auxiliary problems. We provide discussions in Section 6, present related work in Section 7, and conclude in Section 8.

2 NomBank

NomBank annotates the set of arguments of noun predicates, just as PropBank annotates the arguments of verb predicates. As many noun predicates are nominalizations (e.g., *replacement* vs. *replace*), the same *frames* are shared with PropBank as much as possible, thus achieving some consistency with the latter regarding the accepted arguments and the meanings of each label.

Unlike in PropBank, arguments in NomBank can overlap with each other and with the predicate. For example:

> [$_{location}$ U.S.] [$_{pred,subj,obj}$ steelmakers] have supplied the steel.

Here the predicate *make* has subject *steelmakers* and object *steel*, analogous to *Steelmakers make steel.* The difference is that here *make* and *steel* are both part of the word *steelmaker*.

Each argument in NomBank is given one or more labels, out of the following 20: ARG0, ARG1, ARG2, ARG3, ARG4, ARG5, ARG8, ARG9, ARGM-ADV, ARGM-CAU, ARGM-DIR, ARGM-DIS, ARGM-EXT, ARGM-LOC, ARGM-MNR, ARGM-MOD, ARGM-NEG, ARGM-PNC, ARGM-PRD, and ARGM-TMP. Thus, the above sentence is annotated in NomBank as:

> [$_{\text{ARGM-LOC}}$ U.S.] [$_{\text{PRED,ARG0,ARG1}}$ steelmakers] have supplied the steel.

3 Alternating structure optimization

This section gives a brief overview of ASO as implemented in this work. For a more complete description, see (Ando and Zhang, 2005a).

3.1 Multi-task linear classifier

Given a set of training samples consisting of n feature vectors and their corresponding binary labels, $\{\mathbf{X}_i, Y_i\}$ for $i \in \{1, \ldots, n\}$ where each $\mathbf{X}_i$ is a p-dimensional vector, a binary linear classifier attempts to approximate the unknown relation by $Y_i = \mathbf{u}^{\mathrm{T}}\mathbf{X}_i$. The outcome is considered +1 if $\mathbf{u}^{\mathrm{T}}\mathbf{X}$ is positive, or –1 otherwise. A well-established way to find the *weight vector* $\mathbf{u}$ is *empirical risk minimization* with *least square regularization*:

$$\hat{\mathbf{u}} = \arg\min_{\mathbf{u}} \frac{1}{n} \sum_{i=1}^{n} L\big(\mathbf{u}^{\mathrm{T}}\mathbf{X}_i, Y_i\big) + \lambda \|\mathbf{u}\|^2 \quad (1)$$

Function $L(p, y)$ is known as the *loss function*. It encodes the penalty for a given discrepancy between the predicted label and the true label. In this work, we use a modification of Huber's robust loss function, similar to that used in (Ando and Zhang, 2005a):

$$L(p, y) = \begin{cases} -4py & \text{if } py < -1 \\ (1 - py)^2 & \text{if } -1 \le py < 1 \\ 0 & \text{if } py \ge 1 \end{cases} \quad (2)$$

We fix the regularization parameter λ to 10^{-4}, similar to that used in (Ando and Zhang, 2005a). The expression $\|\mathbf{u}\|^2$ is defined as $\sum_{i=1}^{p} \mathbf{u}_p^2$.

When m binary classification problems are to be solved together, a $h \times p$ matrix Θ may be used to capture the *common structures* of the m weight vectors $\mathbf{u}_l$ for $l \in \{1, \ldots, m\}$ $(h \le m)$. We mandate that the rows of Θ be orthonormal, i.e., $\Theta\Theta^{\mathrm{T}} = I_{h \times h}$. The h rows of Θ represent the h most significant components shared by all the $\mathbf{u}$'s. This relationship is modeled by

$$\mathbf{u}_l = \mathbf{w}_l + \Theta^{\mathrm{T}} \mathbf{v}_l \quad (3)$$

The parameters $[\{\mathbf{w}_l, \mathbf{v}_l\}, \Theta]$ may then be found by *joint empirical risk minimization* over all the m problems, i.e., their values should minimize the combined empirical risk:

$$\sum_{l=1}^{m} \left(\frac{1}{n} \sum_{i=1}^{n} L\Big((\mathbf{w}_l + \Theta^{\mathrm{T}} \mathbf{v}_l)^{\mathrm{T}} \mathbf{X}_i^l, Y_i^l \Big) + \lambda \|\mathbf{w}_l\|^2 \right) \quad (4)$$

3.2 The ASO algorithm

An important observation in (Ando and Zhang, 2005a) is that the binary classification problems used to derive Θ are not necessarily those problems we are aiming to solve. In fact, new problems can be invented for the sole purpose of obtaining a better Θ. Thus, we distinguish between two types of problems in ASO: *auxiliary problems*, which are used to obtain Θ, and *target problems*, which are the problems we are aiming to solve[1].

For instance, in the argument identification task, the only target problem is to identify arguments vs. non-arguments, whereas in the argument classification task, there are 20 binary target problems, one to identify each of the 20 labels (ARG0, ARG1, ...).

The target problems can also be used as an auxiliary problem. In addition, we can *invent* new auxiliary problems, e.g., in the argument identification stage, we can predict whether there are three words between the constituent and the predicate using the features of argument identification.

Assuming there are k target problems and m auxiliary problems, it is shown in (Ando and Zhang, 2005a) that by performing one round of minimization, an approximate solution of Θ can be obtained from (4) by the following algorithm:

1. For each of the m *auxiliary problems*, learn $\mathbf{u}_l$ as described by (1).

2. Find $U = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m]$, a $p \times m$ matrix. This is a simplified version of the definition in (Ando and Zhang, 2005a), made possible because the same λ is used for all auxiliary problems.

3. Perform Singular Value Decomposition (SVD) on U: $U = V_1 D V_2^{\mathrm{T}}$, where V_1 is a $p \times m$ matrix. The first h columns of V_1 are stored as rows of Θ.

4. Given Θ, we learn $\mathbf{w}$ and $\mathbf{v}$ for each of the k *target problems* by minimizing the empirical risk of the associated training samples:

$$\frac{1}{n} \sum_{i=1}^{n} L\Big((\mathbf{w} + \Theta^{\mathrm{T}} \mathbf{v})^{\mathrm{T}} \mathbf{X}_i, Y_i \Big) + \lambda \|\mathbf{w}\|^2 \quad (5)$$

5. The weight vector of each target problem can be found by:

$$\mathbf{u} = \mathbf{w} + \Theta^{\mathrm{T}} \mathbf{v} \quad (6)$$

By choosing a convex loss function, e.g., (2), steps 1 and 4 above can be formulated as convex optimization problems and are efficiently solvable.

The procedure above can be considered as a Principal Component Analysis in the predictor space. Step (3) above extracts the most significant components shared by the predictors of the auxiliary problems and hopefully, by the predictors of the target

[1] Note that this definition deviates slightly from the one in (Ando and Zhang, 2005a). We find the definition here more convenient for our subsequent discussion.

problems as well. The hint of potential significant components helps (5) to outperform the simple linear predictor (1).

4 Baseline classifier

The SRL task is typically separated into two stages: argument identification and argument classification. During the identification stage, each constituent in a sentence's parse tree is labeled as either argument or non-argument. During the classification stage, each argument is given one of the 20 possible labels (ARG0, ARG1, . . .). The linear classifier described by (1) is used as the baseline in both stages. For comparison, the F1 scores of a maximum entropy classifier are also reported here.

4.1 Argument identification

Eighteen *baseline features* and six *additional features* are proposed in (Jiang and Ng, 2006) for NomBank argument identification. As the improvement of the F1 score due to the additional features is not statistically significant, we use the set of eighteen baseline features for simplicity. These features are reproduced in Table 1 for easy reference.

1	pred	the stemmed predicate
2	subcat	grammar rule that expands the predicate P's parent
3	ptype	syntactic category (phrase type) of the constituent C
4	hw	syntactic head word of C
5	path	syntactic path from C to P
6	position	whether C is to the left/right of or overlaps with P
7	firstword	first word spanned by C
8	lastword	last word spanned by C
9	lsis.ptype	phrase type of left sister
10	rsis.hw	right sister's head word
11	rsis.hw.pos	POS of right sister's head word
12	parent.ptype	phrase type of parent
13	parent.hw	parent's head word
14	partialpath	path from C to the lowest common ancestor with P
15	ptype & length of path	
16	pred & hw	
17	pred & path	
18	pred & position	

Table 1: Features used in argument identification

Unlike in (Jiang and Ng, 2006), we do not prune arguments dominated by other arguments or those that overlap with the predicate in the training data. Accordingly, we do not maximize the probability of the entire labeled parse tree as in (Toutanova et al., 2005). After the features of every constituent are extracted, each constituent is simply classified independently as either argument or non-argument.

The linear classifier described above is trained on sections 2 to 21 and tested on section 23. A maximum entropy classifier is trained and tested in the same manner. The F1 scores are presented in the first row of Table 3, in columns *linear* and *maxent* respectively. The *J&N* column presents the result reported in (Jiang and Ng, 2006) using both baseline and additional features. The last column *aso* presents the best result from this work, to be explained in Section 5.

4.2 Argument classification

In NomBank, some constituents have more than one label. For simplicity, we always assign exactly one label to each identified argument in this step. For the 0.16% arguments with multiple labels in the training data, we pick the first and discard the rest. (Note that the same is *not* done on the test data.)

A diverse set of 28 features is used in (Jiang and Ng, 2006) for argument classification. In this work, the number of features is pruned to 11, so that we can work with reasonably many auxiliary problems in later experiments with ASO.

To find a smaller set of effective features, we start with all the features considered in (Jiang and Ng, 2006), in (Xue and Palmer, 2004), and various combinations of them, for a total of 52 features. These features are then pruned by the following algorithm:

1. For each feature in the current feature set, do step (2).

2. Remove the selected feature from the feature set. Obtain the F1 score of the remaining features when applied to the argument classification task, on development data section 24 with gold identification.

3. Select the highest of all the scores obtained in

1	position	to the left/right of or overlaps with the predicate
2	ptype	syntactic category (phrase type) of the constituent C
3	firstword	first word spanned by C
4	lastword	last word spanned by C
5	rsis.ptype	phrase type of right sister
6	nomtype	NOM-TYPE of predicate supplied by NOMLEX dictionary
7	predicate & ptype	
8	predicate & lastword	
9	morphed predicate stem & head word	
10	morphed predicate stem & position	
11	nomtype & position	

Table 2: Features used in argument classification

step (2). The corresponding feature is removed from the current feature set if its F1 score is the same as or higher than the F1 score of retaining all features.

4. Repeat steps (1)-(3) until the F1 score starts to drop.

The 11 features so obtained are presented in Table 2. Using these features, a linear classifier and a maximum entropy classifier are trained on sections 2 to 21, and tested on section 23. The F1 scores are presented in the second row of Table 3, in columns *linear* and *maxent* respectively. The *J&N* column presents the result reported in (Jiang and Ng, 2006).

4.3 Further experiments and discussion

In the combined task, we run the identification task with gold parse trees, and then the classification task with the output of the identification task. This way the combined effect of errors from both stages on the final classification output can be assessed. The scores of this complete SRL system are presented in the third row of Table 3.

To test the performance of the combined task on automatic parse trees, we employ two different configurations. First, we train the various classifiers on sections 2 to 21 using gold argument labels and *automatic parse trees* produced by Charniak's reranking parser (Charniak and Johnson, 2005), and test them on section 23 with automatic parse trees. This is the same configuration as reported in (Pradhan et al., 2005; Jiang and Ng, 2006). The scores are presented in the fourth row *auto parse (t&t)* in Table 3.

Next, we train the various classifiers on sections 2 to 21 using gold argument labels and *gold parse trees*. To minimize the discrepancy between gold and automatic parse trees, we remove all the nodes in the gold trees whose POS are *-NONE-*, as they do not span any word and are thus never generated by the automatic parser. The resulting classifiers are then tested on section 23 using automatic parse trees. The scores are presented in the last row *auto parse (test)* of Table 3. We note that *auto parse (test)* consistently outperforms *auto parse (t&t)*.

We believe that *auto parse (test)* is a more realistic setting in which to test the performance of SRL on automatic parse trees. When presented with some previously unseen test data, we are forced to rely on its automatic parse trees. However, for the best results we should take advantage of gold parse trees whenever possible, including those of the *labeled* training data.

	J&N	maxent	linear	aso
identification	82.50	83.58	81.34	85.32
classification	87.80	88.35	87.86	89.17
combined	72.73	75.35	72.63	77.04
auto parse (t&t)	69.14	69.61	67.38	72.11
auto parse (test)	-	71.19	69.05	72.83

Table 3: F1 scores of various classifiers on NomBank SRL

Our maximum entropy classifier consistently outperforms (Jiang and Ng, 2006), which also uses a maximum entropy classifier. The primary difference is that we use a later version of NomBank (September 2006 release vs. September 2005 release). In addition, we use somewhat different features and treat overlapping arguments differently.

5 Applying ASO to SRL

Our ASO classifier uses the same features as the baseline linear classifier. The defining characteristic, and also the major challenge in successfully applying the ASO algorithm is to find related auxiliary problems that can reveal common structures shared with the target problem. To organize our search for good auxiliary problems for SRL, we separate them into two categories, *unobservable auxiliary problems* and *observable auxiliary problems*.

5.1 Unobservable auxiliary problems

Unobservable auxiliary problems are problems whose true outcome cannot be observed from a raw text corpus but must come from another source, e.g., human labeling. For instance, predicting the argument class (i.e., ARG0, ARG1, ...) of a constituent is an unobservable auxiliary problem (which is also the only usable unobservable auxiliary problem here), because the true outcomes (i.e., the argument classes) are only available from human labels annotated in NomBank.

For argument identification, we invent the following 20 binary unobservable auxiliary problems to take advantage of information previously unused at this stage:

> To predict the outcome of argument classification (i.e., ARG0, ARG1, ...) using the features of argument identification (*pred*, *subcat*, ...).

Thus for argument identification, we have 20 auxiliary problems (one auxiliary problem for predicting each of the argument classes ARG0, ARG1, ...) and one target problem (predicting whether a constituent is an argument) for the ASO algorithm described in Section 3.2.

In the argument classification task, the 20 binary target problems are also the unobservable auxiliary problems (one auxiliary problem for predicting each of the argument classes ARG0, ARG1, ...). Thus, we use the same 20 problems as both auxiliary problems and target problems.

We train an ASO classifier on sections 2 to 21 and test it on section 23. With the 20 unobservable auxiliary problems, we obtain the F1 scores reported in the last column of Table 3. In all the experiments, we keep $h = 20$, i.e., all the 20 columns of V_1 are kept.

Comparing the F1 score of ASO against that of the linear classifier in every task (i.e., identification, classification, combined, both auto parse configurations), the improvement achieved by ASO is statistically significant ($p < 0.05$) based on the χ^2 test. Comparing the F1 score of ASO against that of the maximum entropy classifier, the improvement in all but one task (argument classification) is statistically significant ($p < 0.05$). For argument classification, the improvement is not statistically significant ($p = 0.08$).

5.2 Observable auxiliary problems

Observable auxiliary problems are problems whose true outcome can be observed from a raw text corpus without additional externally provided labels. An example is to predict whether *hw=trader* from a constituent's other features, since the head word of a constituent can be obtained from the raw text alone. By definition, an observable auxiliary problem can always be formulated as predicting a feature of the training data. Depending on whether the baseline linear classifier already uses the feature to be predicted, we face two possibilities:

Predicting a used feature In auxiliary problems of this type, we must take care to remove the feature itself from the training data. For example, we must not use the feature *path* or *pred&path* to predict *path* itself.

Predicting an unused feature These auxiliary problems provide information that the classifier was previously unable to incorporate. The desirable characteristics of such a feature are:

1. The feature, although unused, should have been considered for the target problem so it is probably related to the target problem.

2. The feature should not be highly correlated with a used feature, e.g., since the *lastword* feature is used in argument identification, we will not consider predicting *lastword.pos* as an auxiliary problem.

Each chosen feature can create thousands of binary auxiliary problems. E.g., by choosing to predict *hw*, we can create auxiliary problems predicting whether *hw=to*, whether *hw=trader*, etc. To have more positive training samples, we only predict the most frequent features. Thus we will probably predict whether *hw=to*, but not whether *hw=trader*, since *to* occurs more frequently than *trader* as a head word.

5.2.1 Argument identification

In argument identification using gold parse trees, we experiment with predicting three unused features as auxiliary problems: *distance* (distance between the predicate and the constituent), *parent.lsis.hw* (head word of the parent constituent's left sister) and *parent.rsis.hw* (head word of the parent constituent's right sister). We then experiment with predicting four used features: *hw*, *lastword*, *ptype* and *path*.

The ASO classifier is trained on sections 2 to 21, and tested on section 23. Due to the large data size, we are unable to use more than 20 binary auxiliary problems or to experiment with combinations of them. The F1 scores are presented in Table 4.

5.2.2 Argument classification

In argument classification using gold parse trees and gold identification, we experiment with predicting three unused features *path*, *partialpath*, and *chunkseq* (concatenation of the phrase types of text chunks between the predicate and the constituent). We then experiment with predicting three used features *hw*, *lastword*, and *ptype*.

Combinations of these auxiliary problems are also tested. In *all combined*, we use the first 100 problems from each of the six groups of observable auxiliary problems. In *selected combined*, we use the first 100 problems from each of *path*, *chunkseq*, *lastword* and *ptype* problems.

The ASO classifier is trained on sections 2 to 21, and tested on section 23. The F1 scores are shown in Table 5.

feature to be predicted	F1
20 most frequent *distance*s	81.48
20 most frequent *parent.lsis.hw*s	81.51
20 most frequent *parent.rsis.hw*s	81.60
20 most frequent *hw*s	81.40
20 most frequent *lastword*s	81.33
20 most frequent *ptype*s	81.35
20 most frequent *path*s	81.47
linear baseline	81.34

Table 4: F1 scores of ASO with observable auxiliary problems on argument identification. All $h = 20$.

feature to be predicted	F1
300 most frequent *path*s	87.97
300 most frequent *partialpath*s	87.95
300 most frequent *chunkseq*s	88.09
300 most frequent *hw*s	87.93
300 most frequent *lastword*s	88.01
all 63 *ptype*s	88.05
all combined	87.95
selected combined	88.07
linear baseline	87.86

Table 5: F1 scores of ASO with observable auxiliary problems on argument classification. All $h = 100$.

From Table 4 and 5, we observe that although the use of observable auxiliary problems consistently improves the performance of the classifier, the differences are small and not statistically significant. Further experiments combining unobservable and observable auxiliary problems fail to outperform ASO with unobservable auxiliary problems alone.

In summary, our work shows that unobservable auxiliary problems significantly improve the performance of NomBank SRL. In contrast, observable auxiliary problems are not effective.

6 Discussions

Some of our experiments are limited by the extensive computing resources required for a fuller exploration. For instance, "predicting unused features" type of auxiliary problems might hold some hope for further improvement in argument identification, if a larger number of auxiliary problems can be used.

ASO has been demonstrated to be an effective semi-supervised learning algorithm (Ando and Zhang, 2005a; Ando and Zhang, 2005b; Ando, 2006). However, we have been unable to use unlabeled data to improve the accuracy. One possible reason is the cumulative noise from the many cascading steps involved in automatic SRL of unlabeled data: syntactic parse, predicate identification (where we identify nouns with at least one argument), argument identification, and finally argument classification, which reduces the effectiveness of adding unlabeled data using ASO.

7 Related work

Multi-output neural networks learn several tasks simultaneously. In addition to the target outputs, (Caruana, 1997) discusses configurations where both used inputs and unused inputs (due to excessive noise) are utilized as additional outputs. In contrast, our work concerns linear predictors using empirical risk minimization.

A variety of auxiliary problems are tested in (Ando and Zhang, 2005a; Ando and Zhang, 2005b) in the semi-supervised settings, i.e., their auxiliary problems are generated from unlabeled data. This differs significantly from the supervised setting in our work, where only labeled data is used. While (Ando and Zhang, 2005b) uses “predicting used features” (previous/current/next word) as auxiliary problems with good results in named entity recognition, the use of similar observable auxiliary problems in our work gives no statistically significant improvements.

More recently, for the word sense disambiguation (WSD) task, (Ando, 2006) experimented with both supervised and semi-supervised auxiliary problems, although the auxiliary problems she used are different from ours.

8 Conclusion

In this paper, we have presented a novel application of Alternating Structure Optimization (ASO) to the Semantic Role Labeling (SRL) task on NomBank. The possible auxiliary problems are categorized and tested extensively. Our results outperform those reported in (Jiang and Ng, 2006). To the best of our knowledge, we achieve the highest SRL accuracy published to date on the English NomBank.

References

R. K. Ando and T. Zhang. 2005a. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*.

R. K. Ando and T. Zhang. 2005b. A high-performance semi-supervised learning method for text chunking. In *Proc. of ACL*.

R. K. Ando. 2006. Applying alternating structure optimization to word sense disambiguation. In *Proc. of CoNLL*.

C. F. Baker, C. J. Fillmore, and J. B. Lowe. 1998. The Berkeley FrameNet project. In *Proc. of COLING-ACL*.

S. Ben-David and R. Schuller. 2003. Exploiting task relatedness for multiple task learning. In *Proc. of COLT*.

X. Carreras and L. Marquez. 2004. Introduction to the CoNLL-2004 shared task: Semantic role labeling. In *Proc. of CoNLL*.

X. Carreras and L. Marquez. 2005. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proc. of CoNLL*.

R. Caruana. 1997. *Multitask Learning*. Ph.D. thesis, School of Computer Science, CMU.

E. Charniak and M. Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proc. of ACL*.

T. Evgeniou and M. Pontil. 2004. Regularized multitask learning. In *Proc. of KDD*.

D. Gildea and D. Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*.

Z. P. Jiang and H. T. Ng. 2006. Semantic role labeling of NomBank: A maximum entropy approach. In *Proc. of EMNLP*.

K. C. Litkowski. 2004. Senseval-3 task: automatic labeling of semantic roles. In *Proc. of SENSEVAL-3*.

A. Maurer. 2006. Bounds for linear multitask learning. *Journal of Machine Learning Research*.

A. Meyers, R. Reeves, C. Macleod, R. Szekeley, V. Zielinska, B. Young, and R. Grishman. 2004. The NomBank project: An interim report. In *Proc. of HLT/NAACL Workshop on Frontiers in Corpus Annotation*.

C. A. Micchelli and M. Pontil. 2005. Kernels for multitask learning. In *Proc. of NIPS*.

M. Palmer, D. Gildea, and P. Kingsbury. 2005. The Proposition Bank: an annotated corpus of semantic roles. *Computational Linguistics*.

S. S. Pradhan, H. Sun, W. Ward, J. H. Martin, and D. Jurafsky. 2004. Parsing arguments of nominalizations in English and Chinese. In *Proc. of HLT/NAACL*.

S. Pradhan, K. Hacioglu, V. Krugler, W. Ward, J. H. Martin, and D. Jurafsky. 2005. Support vector learning for semantic argument classification. *Machine Learning*.

K. Toutanova, A. Haghighi, and C. D. Manning. 2005. Joint learning improves semantic role labeling. In *Proc. of ACL*.

N. Xue and M. Palmer. 2004. Calibrating features for semantic role labeling. In *Proc. of EMNLP*.

N. Xue. 2006. Semantic role labeling of nominalized predicates in Chinese. In *Proc. of HLT/NAACL*.