Multi-Task Learning in Conditional Random Fields for Chunking in Shallow Semantic Parsing

Saike He[a], Xiaojie Wang[a], Yuan Dong[a,b], Taozheng Zhang[a], and Xue Bai[a]

[a] Bejing University of Posts and Telecommunications, Beijing, 100876, P.R.China
hsk000@gmai1.com, xjwang@bupt.edu.cn, bc003@sina.com
[b] France Telecom R&D Center (Beijing), Beijing, 10080, P.R.China
yuandong@orange-ftgroup.com

Abstract. Alternating Structure Optimization (ASO) is a recently proposed linear Multi-task Learning algorithm. Although its effective has been verified in both semi-supervised as well as supervised methods, yet they necessitate taking external resource as a prerequisite. Therefore, feasibility of employing ASO to further improve the performance merely rests on the labeled data on hand proves to be a task deserving close scrutiny. Catering to this challenging while untapped problem, this paper presents a novel application of ASO to the subtask of Shallow Semantic Parsing: Chunking. Our experiments on Chinese Treebank 5.0 present promising result in chunk analysis, and the error rate is reduced by 5.72%, proposing a profound way to further improve the performance.

Keywords: Shallow Semantic Parsing, Chunking, CRFs, Multi-Task Learning, ASO.

1 Introduction

Semantic parsing of text corpora is needed to support tasks such as information extraction and question-answering. In particular, shallow semantic parsing focuses on identifying the semantic roles of the arguments of a verb (or any predicate) rather than parsing the whole sentence in detail.

Traditional shallow semantic parsing systems for chunk analysis(Kudoh and Matsumoto, 2001; Zhan $et al.$, 2002), which employ machine learning method, focus on the selection of features and their variety combinations to train a statistical model (Carreras and Marquez, 2005). However, we usually have no idea of what a good model is like, neither do we know which features to select. Fortunately, Multi-task Learning (MTL) provides us with a substantial solution to work out this nontrivial problem(Caruana, 1997; Ben-David and Schuller, 2003; Evgeniou and Pontil, 2004; Miccheli and Pontil, 2005; Maurer, 2006). Multi-task Learning is an approach to inductive transfer that emphasizes learning multiple tasks in parallel while using a shared representation, so that what is learned by all tasks is available to the target task.

Alternating Structure Optimization (ASO) algorithm (Ando and Zhang, 2005a) is a linear method based on such idea with the shared representation projected as a low-dimensional projection matrix shared by all related tasks. In supervised learning applications, large amount of unlabeled data is readily available, while labeled data are costly to obtain. Therefore, (Ando and Zhang, 2005b) employs ASO in semi-supervised method and seeks to discover shared predictive structures through jointly learning multiple classification problems on unlabeled data. As adequate unlabeled data is involved, promising result achieved.

Based on the consideration of huge resource consumption in (Ando and Zhang, 2005b) and the availability of large human-labeled corpora such as PropBank (Palmer *et al.*, 2005) and FrameNet (Baker *et al.*, 1998), (Liu and Ng, 2007) utilizes labeled data to train ASO classifier, reasonably promising performance is obtained with comparably less additional labeled corpora.

Although ASO proves to be efficient in both semi-supervised and supervised methods as aforementioned, yet external resources (labeled or unlabeled data) necessitate to be provided to guarantee the effectiveness of ASO. Therefore, whether we can further improve the performance merely rest on the labeled data on hand turns out to be a challenging yet untapped problem.

Our research have been focused on developing a general learning framework for employing ASO on the given labeled data to further boost the performance of chunk analysis in shallow semantic parsing. We employ the basic idea of Feature Nets (Caruana, 1997) in our chunking system, which utilizes Conditional Random Fields (CRFs)(Lafferty *et al.*, 2001) as the basic statistical model.

Experiment results of chunking on Chinese Treebank 5.0[1] present an error reduction of 5.72%, demonstrating the effectiveness of this strategy. It is worth noting that this result is obtained without any additional resource compared with that used in (Ando and Zhang, 2005b) and (Liu and Ng, 2007).

This paper enjoys contribution in tri-fold. First, we make reasonable modification on the original version of ASO, the memory and time consumption is greatly reduced. Second, we incorporated ASO into CRFs through Feature Nets, promising results achieved in contrast to its simplicity. Finally, a fundamental conclusion has been drawn on the key point of improving the performance of a combined system.

The rest of the paper is organized as follows: Section 2 describes the framework of our chunking system briefly. Section 3 introduces ASO algorithm and its modification. Section 4 describes a combined model with ASO in detail. Section 5 presents and analyzes our experimental results, followed by an in-depth discussion on 2 worthy issues in Section 6. Finally, we conclude the work in Section 7.

2 Framework of Shallow Semantic Parsing

Our framework for chunking system utilizes CRFs as the basic statistical model. The features used to train CRFs will also be introduced briefly in this section.

2.1 Conditional Random Fields (CRFs)

CRFs for sequence labeling offers advantages over both generative models like HMMs and classifiers applied at each sequence position. CRFs is undirected graph established on $G = (V, E)$, where V is the set of random variables $Y = \{Y_i | 1 \leq i \leq n\}$ for each the n tokens in an input sequence and $E = \{(Y_{i-1}, Y_i) | 1 \leq i \leq n\}$ is the set of $(n-1)$ edges forming a linear chain. Following (Lafferty *et al.*, 2001), Let $\mathbf{O} = \{o_1, o_2, ..., o_T\}$ denote some observed input data sequences such as a sequence of phrases in training data. Let $\mathbf{S} = \{s_1, s_2, ..., s_T\}$ be a set of finite state machine (FSM) states, each of which is associated with a label (such as SBJ, OBJ , LOC, TMP in chunk analysis), then the conditional probability of the state sequence S given the input sequence **O** is computed as follows:

$$P_\Lambda(\mathbf{S}|\mathbf{O}) = \frac{1}{Z_O} \prod_{c=C(\mathbf{S},\mathbf{O})} exp\left(\sum_{t=1}^{T}\sum_{k=1}^{K} \lambda_k f_k\left(s_{t-1}, s_t, o, t\right)\right) \tag{1}$$

where f_k is an arbitrary feature function; and λ_k is the weight for each feature function; it can be optimized through iterative algorithms like GIS (Darroch and Ratcliff, 1972). Recent research indicates that quasi-Newton methods such as L-BFGS (Byrd *et al.*, 1994) are more effective than GIS.

[1] A detailed description of our data set could be found at:
http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2005T01

2.2 Feature Templates

Table 1 lists the features we used in our chunking system to train CRFs, and they are grouped into 4 categories: features based on word, features based on pos, compound features (involving both word and pos), and finally, features based on position. The ‘position feature’ takes 2 values: *beforeVP* and *afterVP*, this is used to distinguish ssubjective and objective. In general, subjective locates on the left side of a verb while objective locates on its right side.

Table 1: Features used in the chunking system.

	Type	**Feature**
Word	Unigram	$W_n(n=-2,-1,0,1,2)$
	Bigram	$W_nW_{n+1}(n=-2,-1,0,1)$
	Trigram	$W_{n+1}W_nW_{n-1}(n=-1,0,1)$
	Jump	$W_{-1}W_1$
POS	Unigram	$P_n(n=-2,-1,0,1,2)$
	Bigram	$P_nP_{n+1}(n=-2,-1,0,1)$
	Trigram	$P_{n+1}P_nP_{n-1}(n=-1,0,1)$
	Jump	$P_{-1}P_1$
Compound	Bigram	$W_nP_n(n=,-1,0,1)$
	Trigram	$W_0P_{n+1}P_n(n=-2,1)$
	Jump	$P_{-1}W_0P_1$
Position	Unigram	$beforeVP/afterVP$

Note:W and P represent word and pos respectively; subscript n indicates the relative position taking the current word as its reference.

3 Multi-Task Learning

In statistics domain, most tasks are trained on a single, isolated, difficult task, rendering its paleness to learn the model well. It might be better to require the learner to learn many tasks simultaneously. If the tasks can share what they learn, the learner may find it is easier to learn them together than in isolation. We call this approach Multi-task Learning (MTL) (Caruana, 1997). The task we wish to learn better is the **target task**. The related tasks whose training signals are used by Multi-task Learning to learn the target task better are **auxiliary tasks**.

Multitask Learning could be efficiently cast as a joint optimization problem (Ando and Zhang, 2005a), and can be solved by alternating structure optimization (ASO) algorithm.

In the following section, we will present a brief overview of ASO. For a more complete description, pleas refer to (Ando and Zhang, 2005a). Then, we will introduce our modification on ASO, which reduces the resource consumption as well as the running time.

3.1 The ASO algorithm

In the literature of traditional single task learning (also called STL) (Caruana, 1997), for a given finite set of training samples $\{X_i, Y_i\}(i = 1 \dots n)$, the formulation of linear predicator is given by:

$$f(x) = \mathbf{u}^T X \tag{2}$$

where **u** is weight vector. The frequently used method for finding a optimum **u** is to minimize the empirical error on the training data (ERM):

$$\widehat{\mathbf{u}} = \arg\min_{\mathbf{u}} \frac{1}{n} \sum_{i=1}^{n} L\left(\mathbf{u}^T\mathbf{X}, Y_i\right) + \lambda \|\mathbf{u}\|^2 \tag{3}$$

Function $L(p, y)$ is referred to as loss function. It encodes the penalty for the disparity between the predicted label and the gold label. The formulation of L(p,y) is given below:

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

Letting $q = py$, we could obtain an equivalent definition of the loss function L as in formula (4):

$$L(p, y) = M(q) = \begin{cases} -4q & \text{if } q < -1 \\ (1 - q)^2 & \text{if } -1 \leq q < 1 \\ 0 & \text{if } q \geq -1 \end{cases} \tag{5}$$

The regularization parameter λ is fixed to 10^{-4}. The formulation of loss function given in formula (5) is for the convenient of our further modification, which could be seen in Section 3.2.

Under the circumstance of MTL, there is assumed to be a low-dimensional projection matrix $\Theta(h \times p)$ shared by all related tasks and used to capture the common structures of the m weight vectors $\mathbf{u}_l$for$l \in \{1, \ldots, m\}(h \leq m)$. The h rows of Θ represent the h most significant components shared by all the u, which for task l is modeled by

$$\mathbf{u}_l = \mathbf{w}_l + \Theta^T \mathbf{v}_l \tag{6}$$

The optimum parameters $[\{w_l, v_l\}]$ could be found by joint empirical risk minimization over all the m problems:

$$\sum_{l=1}^{m} \left(\frac{1}{n} \sum_{i=1}^{n} L\left(\left(\mathbf{w}_l + \Theta^T \mathbf{v}_l \right)^T \mathbf{X}_i^l, \mathbf{Y}_i^l \right) + \lambda \|\mathbf{w}_l\|^2 \right) \tag{7}$$

3.2 Modification on ASO

Although ASO algorithm introduced by (Ando and Zhang, 2005a) is reasonable, yet it is not perfect. Since Θ is obtained by processing all the auxiliary tasks simultaneously, thus great memory space and training time is needed. Hence, it is not always tractably especially when extremely large auxiliary tasks are involved in. Fortunately, a way work this problem out exists and is enlightened by the work of (Liu and Ng, 2007).

This section describes in detail the mathematics of ASO. We rewrite the optimization problems (7) in matrix form as shown below. We first make equal conversion of (7) to obtain:

$$\widehat{\mathbf{w}}, \widehat{\mathbf{v}} = \arg\min_{\mathbf{w,v}} \frac{1}{n} \sum_{i=1}^{n} L\left(\left(\mathbf{w} + \Theta^T \mathbf{v} \right)^T \mathbf{X}, Y_i \right) + \lambda \|\mathbf{u}\|^2 \tag{8}$$

We define a vector version of the single-parameter robust loss function M:

$$\mathbf{M(Q)} = \left(M\left(Q_1\right), M\left(Q_2\right), \ldots, M\left(Q_p\right) \right)^T \tag{9}$$

Thus, the linear classifier optimization (3) can be rewritten as:

$$\begin{aligned}\widehat{\mathbf{u}} &= \arg\min_{\mathbf{u}} \frac{1}{n}\sum_{i=1}^{n} L\left(\mathbf{u}^T\mathbf{X}_i, Y_i\right) + \lambda\|\mathbf{u}\|^2 \\ &= \arg\min_{\mathbf{u}} \frac{1}{n}\sum_{i=1}^{n} L\left(\mathbf{X}_i^T\mathbf{u}, Y_i\right) + \lambda\|\mathbf{u}\|^2 \\ &= \arg\min_{\mathbf{u}} \frac{1}{n} sum\left(\mathbf{M}\left(X\mathbf{u} \bullet \mathbf{Y}\right)\right) + \lambda\|\mathbf{u}\|^2\end{aligned} \tag{10}$$

The *sum()* function calculates the sum of all elements of a vector. The operator $\bullet$ represents pointwise vector multiplication. Define $\mathbf{P} = X\mathbf{u}, \mathbf{Y} = (\mathbf{Y} \bullet X)\,\mathbf{u}$, the function value supplied to L-BFGS is then:

$$\widehat{\mathbf{u}} = \frac{1}{n} sum\left(\mathbf{M}\left(\mathbf{P}\right)\right) + \lambda\|\mathbf{u}\|^2 \tag{11}$$

In order to obtain the gradient, we need differentiate the loss function:

$$\frac{dM(q)}{dq} = \begin{cases} -4 & \text{if } q < -1 \\ 2(q-1) & \text{if } -1 \le q < 1 \\ 0 & \text{if } q \ge -1 \end{cases} \tag{12}$$

A vector of gradient is defined as:

$$\mathbf{N}\left(\mathbf{Q}\right) = \frac{d}{d\mathbf{Q}} = \left(\left.\frac{dM(q)}{dq}\right|_{\mathbf{Q1}}, \left.\frac{dM(q)}{dq}\right|_{\mathbf{Q2}}, \ldots, \left.\frac{dM(q)}{dq}\right|_{\mathbf{Qp}} \right)^T \tag{13}$$

And the gradient is:

$$\begin{aligned}\frac{df(\mathbf{u})}{d\mathbf{u}} &== \frac{1}{n}\frac{d(\mathbf{u})}{d\mathbf{u}} sum\left(\mathbf{M}\left(\mathbf{P}\right)\right) + \lambda\frac{d(\mathbf{u})}{d\mathbf{u}}\|\mathbf{u}\|^2 \\ &= \frac{1}{n}\left(\frac{d(\mathbf{P})}{d\mathbf{u}}\right)^T \frac{d}{d\mathbf{P}} sum\left(\mathbf{M}\left(\mathbf{P}\right)\right) + 2\lambda\mathbf{u} \\ &= \frac{1}{n}\left(\mathbf{Y} \bullet X\right)^T \mathbf{N}\left(\mathbf{P}\right) + 2\lambda\mathbf{u} \\ &= \frac{1}{n} X^T\left(\mathbf{Y} \bullet \mathbf{N}\left(\mathbf{P}\right)\right) + 2\lambda\mathbf{u}\end{aligned} \tag{14}$$

The common structure are represented by the h rows of Θ. We further define:

$$\mathbf{u} = \mathbf{w} + \Theta^T\mathbf{v} \tag{15}$$

The optimization (7) can now be rewritten in matrix form.

$$\begin{aligned}\widehat{\mathbf{w}}, \widehat{\mathbf{v}} &= \arg\min_{\mathbf{w,v}} \frac{1}{n}\sum_{i=1}^{n} L\left(\left(\mathbf{w} + \Theta^T\mathbf{v}\right)^T \mathbf{X}, Y_i\right) + \lambda\|\mathbf{u}\|^2 \\ &= \arg\min_{\mathbf{w,v}} \frac{1}{n}\sum_{i=1}^{n} L\left(\mathbf{u}^T\mathbf{X}_i, Y_i\right) + \lambda\|\mathbf{w}\|^2 \\ &= \arg\min_{\mathbf{w,v}} \frac{1}{n} sum\left(\mathbf{M}\left(X\mathbf{u} \bullet \mathbf{Y}\right)\right) + \lambda\|\mathbf{w}\|^2\end{aligned} \tag{16}$$

We can now conclude:

$$f\left(\mathbf{w},\mathbf{v}\right)=\frac{1}{n}sum\left(\mathbf{M}\left(\mathbf{P}\right)\right)+\lambda\|\mathbf{w}\|^2 \tag{17}$$

where $\mathbf{P}=X\left(\mathbf{w}+\Theta^T\mathbf{v}\right)\bullet\mathbf{Y}$. And the gradients are:

$$\begin{aligned}
\frac{df\left(\mathbf{w},\mathbf{v}\right)}{d\mathbf{w}}&=\frac{1}{n}\frac{d(\mathbf{u})}{d\mathbf{w}}^T\frac{d}{d\mathbf{u}}sum\left(\mathbf{M}\left(\mathbf{P}\right)\right)+\lambda\frac{d}{d\mathbf{w}}\|\mathbf{w}\|^2\\
&=\frac{1}{n}X^T\left(\mathbf{Y}\bullet\mathbf{N}\left(\mathbf{P}\right)\right)+2\lambda\mathbf{w}\\
\frac{df\left(\mathbf{w},\mathbf{v}\right)}{d\mathbf{v}}&=\frac{1}{n}\frac{d(\mathbf{u})}{d\mathbf{v}}^T\frac{d}{d\mathbf{u}}sum\left(\mathbf{M}\left(\mathbf{P}\right)\right)+\lambda\frac{d}{d\mathbf{v}}\|\mathbf{w}\|^2\\
&=\frac{1}{n}\Theta X^T\left(\mathbf{Y}\bullet\mathbf{N}\left(\mathbf{P}\right)\right)
\end{aligned} \tag{18}$$

Our contribution of the former modification lies in 2 aspects. First, the **u** used to obtain Θ can now be trained in isolation, thus memory consumption is reduced greatly. Secondly, as each **u** could be trained separately, it makes multi-thread mechanism feasible. As the training process could be conducted on several CPUs at the same time, thus the time consumption is nearly inversely proportional to the number of CPUs.

3.3 Target Task and Auxiliary Tasks

Having nailed down the algorithm for ASO, we now turn to the construction of target task and auxiliary tasks. We choose a classifier trained on the original training data to make inference for chunking prediction as our target task, while the auxiliary tasks are obtained by the following rule: dividing the training data into several parts, then each part could be used to train a chunking predictor, thus constituting an auxiliary task. Notice, under the current circumstance, the target task and each auxiliary task are essentially same tasks except for their data sizes. This might be different from the rules used by (Ando and Zhang, 2005b) or (Liu and Ng, 2007) in constructing auxiliary tasks. As great amount of additional data is involved in both (Ando and Zhang, 2005b) and (Liu and Ng, 2007), their method for obtaining auxiliary tasks could meet the statistical property while data sparsity is not so serious. That is to say, their adequate data size could compensate the irrelevancy between target task and auxiliary tasks. In contrast, only limited data is presumed in our experiment setting. Therefore, high relevance between the target task and auxiliary tasks is necessary to guarantee the final performance. Our strategy mentioned above could not only preserves the mechanism of MTL, but also proves effective. This will be shown later in Section 5.

4 ASO in CRFs Model

To boost the final performance by utilizing ASO, our final system for chunking in shallow semantic parsing is implemented in two steps. In the first step, we need to train an ASO classifier. The features we use are the same as that in Table 1. Sparked by the basic idea of Feature Nets, in the second step, we decode the testing data with ASO classifier, then encode the predicted chunking information into additional features (hereafter, ASO features), indicating the potential chunk label for the word currently considered. As for the training data, the ‘ASO features’ are directly obtained by the gold-standard chunk labels, that is to say, using the gold labels as ASO features. With features defined in Table 1 and Table 2, a CRFs model could be trained and then tested, each phrase with ASO features involved. The flow chart of our combined system is shown in Figure 1.

As ASO is derived from the mechanism of MTL, it could dig out novel features that would never be discovered in STL mode. Though there might unavoidably exist some errors in the output of ASO, however, a model of STL could benefit greatly from the correct portion as the errors could be masked through feature combination. That is to say, the attempt of utilizing ASO features alone

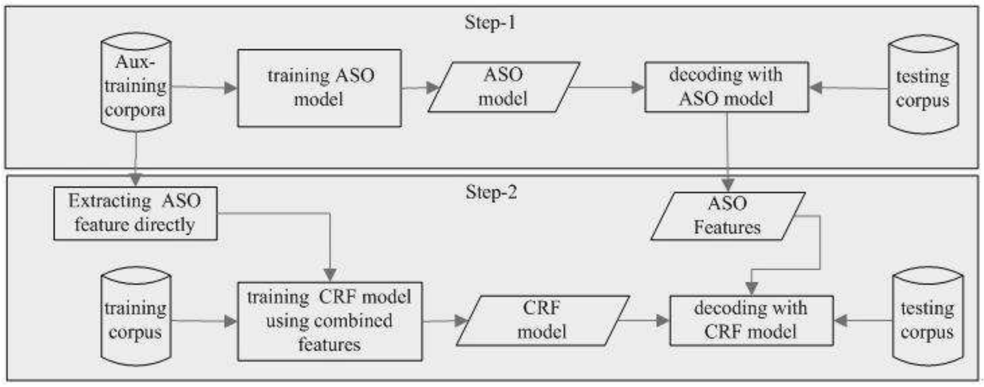

Figure 1: Flow chart of the combined system.

would deteriorate the overall performance as these errors might mislead the training of CRFs model. Thus, a more wise strategy would be masking the errors and filtering out the beneficial part through feature combination. Thereafter, only compound ASO features are involved in the final combination system, which could be found in Table 2.

Table 2: Compound ASO features.

	Type	Feature
ASO Compound	Bigram	$W_nA_n(n=-1,0,1)$
		$P_nA_n(n=-2,-1,0,1)$
	Trigram	$W_nP_nA_n(n=-1,0,1)$
	Jump	$A_{-1}P_0A_1$

Where A represents additional feature derived from ASO.

Note: Only combined features associating with ASO features are list here. The other parts of features also utilized in the final combination system are listed in Table 1.

5 Evaluation Results

This section reports the experiment results in chunking based on the Chinese Treebank 5.0, of which $chtb\_001.fid \sim chtb\_300.fid$ are the training data, while $chtb\_301.fid \sim chtb\_400.fid$ are the testing data. In our experiment, we mainly consider the performance on SBJ, OBJ, TMP and LOC for chunking analysis. The performance is measured by F-score, $F = 2RP/(R+P)$, where the recall R and precision P are respectively the proportions of the correctly labeled chunks to all the chunks in the gold-standard label and the model's output.

5.1 Baseline System

For comparison, we also trained 2 traditional classifiers in STL mode using the same features as that in our combined system.

Linear Classifier: Theoretically, it is a kind of STL method. In implementation, for the sake of convenience, we still employ our ASO toolkit to training this linear classifier. Here, no auxiliary tasks are involved, only the training data is trained in isolation, therefore the ostensibly MTL method is actually in STL mode, thus guarantees the correctness of this method. This linear classifier is involved as an ancillary comparison in justify the superiority of ASO in MTL mode.

SVM Classifier: Support Vector Machines[2](SVMs, which is also a statistical learner in STL), first introduced by Vapnik (Cortes and Vapnik, 1995; Vapnik, 1995), are relatively effective ap-

[2] We use the Support Vector Machines Toolkit from: http://svmlight.joachims.org/.

proaches for solving two-class pattern recognition problems. For its well-know good generalization performance and ability to tackle multi-classes problems, SVMs have been applied to a myriad of fields in NLP. Here, SVMs is involved to clarify what is the key factor that determines the final performance of a combined system, as well as certify whether ASO model with a relative lower performance could bring more benefit than the one with a higher performance in STL mode.

5.2 Experiment Setting

Under the circumstance of our experiments setting, the auxiliary tasks are created following the rules given below: the whole training data (which is $chtb\_001.fid \sim chtb\_150.fid$) are used as the target task, then this training data is divided equally into 7 parts to constitute the auxiliary tasks, namely, aux_task_1, aux_task_2 … aux_task_7. In order to guarantee the reliability of the experiment result, 8-fold cross validation has been conducted.

5.3 Experiment Results

Table 3 shows the gradually change in performance as auxiliary tasks are involved into ASO model. Note, STL(linear classifier) switches to MTL once the first auxiliary task is involved.

Table 3: Comparison of ASO with linear classifier.

Run ID		P	R	F
STL	Linear	64.26	53.05	58.12
MTL (ASO)	+Aux_Task_1	64.17	53.95	58.62
	+Aux_Task_2	64.22	53.96	58.65
	+Aux_Task_3	64.20	54.10	58.72
	+Aux_Task_4	64.19	54.15	58.74
	+Aux_Task_5	64.25	54.65	59.06
	+Aux_Task_6	64.20	54.89	59.18
	+Aux_Task_7	64.19	54.98	59.23

As shown in Table 3, compared with the linear model in STL mode, ASO improves the recall while degrading the precision, ultimately, the final F-score is lifted to a higher plateau: 59.23%, with the error rate reduced the by 2.65%, thus demonstrating the effectiveness of the strategy for constructing auxiliary tasks as introduced in Section 3.3.

Except for the relative improvement, what we do care about is what fraction of extra information ASO can provide for a stronger STL learner to benefit from. This will be shown in Table 4, while a more detailed discussion will be presented in Section 6.

Table 4: Performance comparison of CRFs with systems in different configurations.

Run ID		P	R	F
STL	Linear	64.26	53.05	58.12
	SVMs	85.68	86.47	86.07
MTL	ASO	64.19	54.98	59.23
BaseLine	CRFs	92.41	93.67	93.04
(Combined) (Systems)	CRFs + Linear	91.15	92.36	91.75
	CRFs + SVMs	92.26	93.32	92.79
	CRFs + ASO	92.21	94.70	**93.43**

Table 4 shows the performance comparison of CRFs[3] with systems in different configurations. In this experiment group, CRFs model is the baseline system, and we investigate the optimal com-

[3] We use its implementation CRF++ by Taku Kudo, which can be downloaded from: http://crfpp.sourceforge.net/

bination of it with other models. Here, SVMs are involved and serve as a competent comparison to justify whether a higher performance leaner is certainly more beneficial than an inferior one.

As shown in Table 4, CRFs combined with ASO achieves the highest performance, 93.43, reducing the total error rate by 5.72%, when taking CRFs alone as a baseline. On the other side, neither Linear nor SVMs classifier bring any benefit when combined with CRFs.

6 Discussion

6.1 ASO Compared with Linear Predictor

As aforementioned in Sectioned 5.3, compared with linear predictor, the improved performance of ASO is mainly due to the lifted recall rate while the precision is actually degraded. However, this is inconsistent with the conclusion drawn in (Ando and Zhang, 2005b) that ASO could improve both precision and recall. There might be a dual explanation. First, the data sets we utilized are different in both the language type and size. Secondly, which may proves to be more crucial: the experiments conducted in (Ando and Zhang, 2005b) is based on semi-supervised learning, that is to say, a large amount of unlabeled data are involved to capture the common law inherent in the huge data set, which is then encoded in a common structure. Thus, its effectiveness is based on information from a broad data span, which would be beneficial both for the precision and recall. However, this is not the case in our experiment setting. As none of any additional data are involved, the only additional information is obtained through digging out information in depth based on the given data. This two aspects may function together to contribute to such disparity.

6.2 The Effectiveness of CRFs Combined with ASO

Based on the result from Table 4, we draw the conclusion that only CRFs combined with ASO bring about performance promotion. The conception of **'Information Coverage Rate (ICR)'** may serve as an adequate explanation to this phenomenon. For the sake of measurement convenience, here the ICR is defined as the ratio of chunks correctly recognized by statistic model to the gold-standard chunks. The ICR information of each model is given in Table 5.

Table 5: Information Coverage Rate of each model.

Target Object	F	ICR
CRFs	93.04	93.67
CRFs + Linear	91.75	93.74
CRFs + SVMs	92.79	93.95
CRFs + ASO	93.43	96.33

Note: here '+' does not mean a combined model, it merely indicates that the 2 models involved are just grouped together to calculate ICR.

Table 5 presents the statistical information in the form of ICR. On one hand, though ASO in isolation performs only slightly better than linear classifier in F-score, as shown in Table 4, the additional information it could provide is really large, contributing an ICR value of 96.33%. It is this large portion of extra information that guarantees the superiority in the combined model. On the other hand, the relative lower ICR of 'SVMs + CRFs' could explain its inferior to the combined CRFs with ASO. Since the additional correct information is already limited, the inaccurate part outweigh the limited benefit, these two factors play a dual function and ultimately port detriment to the final performance.

7 Conclusion

Casting to the challenging problem of further improve performance merely based on given data without any external resource, we presented a novel strategy to employ ASO in supervised learning. Experiments of chunking in shallow semantic parsing on Chinese Treebank 5.0 indicate the effectiveness of our strategy, with the error rate reduced by 5.72%. Close scrutiny to our experiments indicates 2 aspects worthy mention: (1) As the information provide by ASO is mixed with errors to some extent, the ASO feature should be utilized delicately. In order to mask the misleading information, ASO feature should be used in compound features only. (2) To justify whether a given model would bring benefit when combined with another, the key lies in the fraction of additional information it can provide. In other words, high performance model does not necessitate guaranteeing high performance in the combined model.

References

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

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

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

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

Byrd, R.H., J. Nocedal and R. .B. Schnabel. 1994. Representations of quasi-Newton matrices and their use in limited memory methods. *Mathematical Programming*, (63): 129-156.

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

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

Cortes, C. and Vladimir N. Vapnik. 1995. Support Vector Networks. *Machine Learning*, 20:273-297.

Darroch, J.N. and D. Ratcliff. 1972. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, 43 (5):1470-1480.

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

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

Lafferty, J., A. McCallum and F. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proc. of the 18th ICML*, pp.282-289, San Francisco, CA.

Liu, Chang and Hwee Tou Ng. 2007. Learning Predictive Structures for Semantic Role Labeling of NomBank. *Proc. of the 45th Annual Meeting of the ACL*.

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

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

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

Vapnik, Vladimir N. 2005. *The Nature of Statistical Learning Theory*. Springer.

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