Semantic Role Labeling via Integer Linear Programming Inference

Vasin Punyakanok Dan Roth Wen-tau Yih Dav Zimak
Department of Computer Science
University of Illinois at Urbana-Champaign
{punyakan,danr,yih,davzimak}@uiuc.edu

Abstract

We present a system for the semantic role labeling task. The system combines a machine learning technique with an inference procedure based on integer linear programming that supports the incorporation of linguistic and structural constraints into the decision process. The system is tested on the data provided in CoNLL-2004 shared task on semantic role labeling and achieves very competitive results.

1 Introduction

Semantic parsing of sentences is believed to be an important task toward natural language understanding, and has immediate applications in tasks such information extraction and question answering. We study *semantic role labeling(SRL)*. For each verb in a sentence, the goal is to identify all constituents that fill a semantic role, and to determine their roles, such as Agent, Patient or Instrument, and their adjuncts, such as Locative, Temporal or Manner.

The PropBank project (Kingsbury and Palmer, 2002) provides a large human-annotated corpus of semantic verb-argument relations. Specifically, we use the data provided in the CoNLL-2004 shared task of semantic-role labeling (Carreras and Màrquez, 2003) which consists of a portion of the PropBank corpus, allowing us to compare the performance of our approach with other systems.

Previous approaches to the SRL task have made use of a full syntactic parse of the sentence in order to define argument boundaries and to determine the role labels (Gildea and Palmer, 2002; Chen and Rambow, 2003; Gildea and Hockenmaier, 2003; Pradhan et al., 2003; Pradhan et al., 2004; Surdeanu et al., 2003). In this work, following the CoNLL-2004 shared task definition, we assume that the SRL system takes as input only partial syntactic information, and no external lexico-semantic knowledge bases. Specifically, we assume as input resources a part-of-speech tagger, a shallow parser that can process the input to the level of based chunks and clauses (Tjong Kim Sang and Buchholz, 2000; Tjong Kim Sang and Déjean, 2001), and a named-entity recognizer (Tjong Kim Sang and De Meulder, 2003). We do *not* assume a full parse as input.

SRL is a difficult task, and one cannot expect high levels of performance from either purely manual classifiers or purely learned classifiers. Rather, supplemental linguistic information must be used to support and correct a learning system. So far, machine learning approaches to SRL have incorporated linguistic information only implicitly, via the classifiers' features. The key innovation in our approach is the development of a principled method to combine machine learning techniques with linguistic and structural constraints by explicitly incorporating inference into the decision process.

In the machine learning part, the system we present here is composed of two phases. First, a set of argument candidates is produced using two learned classifiers—one to discover beginning positions and one to discover end positions of each argument type. Hopefully, this phase discovers a small superset of all arguments in the sentence (for each verb). In a second learning phase, the candidate arguments from the first phase are re-scored using a classifier designed to determine argument type, given a candidate argument.

Unfortunately, it is difficult to utilize global properties of the sentence into the learning phases. However, the inference level it is possible to incorporate the fact that the set of possible role-labelings is restricted by both structural and linguistic constraints—for example, arguments cannot structurally overlap, or, given a predicate, some argument structures are illegal. The overall decision problem must produce an outcome that consistent with these constraints. We encode the constraints as linear inequalities, and use integer linear programming(ILP) as an inference procedure to make a final decision that is both consistent with the constraints and most likely according to the learning system. Although ILP is generally a computationally hard problem, there are efficient implementa-

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
2004		**00-00-2004 to 00-00-2004**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Semantic role Labeling via Integer Linear Programming Inference	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of Illinois,Department of Computer Science ,Urbana,IL,61801**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**7**	

tions that can run on thousands of variables and constraints. In our experiments, we used the commercial ILP package (Xpress-MP, 2003), and were able to process roughly twenty sentences per second.

2 Task Description

The goal of the semantic-role labeling task is to discover the verb-argument structure for a given input sentence. For example, given a sentence " I *left* my pearls to my daughter-in-law in my will", the goal is to identify different arguments of the verb *left* which yields the output:

> [A0 I] [V *left*] [A1 my pearls] [A2 to my daughter-in-law] [AM-LOC in my will].

Here A0 represents the *leaver*, A1 represents the *thing left*, A2 represents the *benefactor*, AM-LOC is an adjunct indicating the location of the action, and V determines the verb.

Following the definition of the PropBank, and CoNLL-2004 shared task, there are six different types of arguments labelled as A0-A5 and AA. These labels have different semantics for each verb as specified in the PropBank Frame files. In addition, there are also 13 types of adjuncts labelled as AM-XXX where XXX specifies the adjunct type. In some cases, an argument may span over different parts of a sentence, the label C-XXX is used to specify the continuity of the arguments, as shown in the example below.

> [A1 The pearls] , [A0 I] [V *said*] , [C-A1 were left to my daughter-in-law].

Moreover in some cases, an argument might be a relative pronoun that in fact refers to the actual agent outside the clause. In this case, the actual agent is labeled as the appropriate argument type, XXX, while the relative pronoun is instead labeled as R-XXX. For example,

> [A1 The pearls] [R-A1 which] [A0 I] [V *left*] , [A2 to my daughter-in-law] are fake.

See the details of the definition in Kingsbury and Palmer (2002) and Carreras and Màrquez (2003).

3 System Architecture

Our semantic role labeling system consists of two phases. The first phase finds a subset of arguments from all possible candidates. The goal here is to filter out as many as possible false argument candidates, while still maintaining high recall. The second phase focuses on identifying the types of those argument candidates. Since the number of candidates is much fewer, the second phase is able to use slightly complicated features to facilitate learning a better classifier. This section first introduces the learning system we use and then describes how we learn the classifiers in these two phases.

3.1 SNoW Learning Architecture

The learning algorithm used is a variation of the Winnow update rule incorporated in SNoW (Roth, 1998; Roth and Yih, 2002), a multi-class classifier that is specifically tailored for large scale learning tasks. SNoW learns a sparse network of linear functions, in which the targets (argument border predictions or argument type predictions, in this case) are represented as linear functions over a common feature space. It incorporates several improvements over the basic Winnow multiplicative update rule. In particular, a regularization term is added, which has the effect of trying to separate the data with a thick separator (Grove and Roth, 2001; Hang et al., 2002). In the work presented here we use this regularization with a fixed parameter.

Experimental evidence has shown that SNoW activations are monotonic with the confidence in the prediction. Therefore, it can provide a good source of probability estimation. We use softmax (Bishop, 1995) over the raw activation values as conditional probabilities, and also the score of the target. Specifically, suppose the number of classes is n, and the raw activation values of class i is act_i. The posterior estimation for class i is derived by the following equation.

$$\text{score}(i) = p_i = \frac{e^{act_i}}{\sum_{1 \le j \le n} e^{act_j}}$$

The score plays an important role in different places. For example, the first phase uses the scores to decide which argument candidates should be filtered out. Also, the scores output by the second-phase classifier are used in the inference procedure to reason for the best global labeling.

3.2 First Phase: Find Argument Candidates

The first phase is to predict the argument candidates of a given sentence that correspond to the active verb. Unfortunately, it turns out that it is difficult to predict the exact arguments accurately. Therefore, the goal here is to output a superset of the correct arguments by filtering out unlikely candidates.

Specifically, we learn two classifiers, one to detect beginning argument locations and the other to detect end argument locations. Each multi-class classifier makes predictions over forty-three classes—thirty-two argument types, ten continuous

argument types, and one class to detect *not beginning/not end*. Features used for these classifiers are:

- **Word** feature includes the current word, two words before and two words after.
- **Part-of-speech tag** (POS) feature includes the POS tags of all words in a window of size two.
- **Chunk** feature includes the BIO tags for chunks of all words in a window of size two.
- **Predicate lemma & POS tag** show the lemma form and POS tag of the active predicate.
- **Voice** feature is the voice (active/passive) of the current predicate. This is extracted with a simple rule: a verb is identified as passive if it follows a to-be verb in the same phrase chunk and its POS tag is VBN(past participle) or it immediately follows a noun phrase.
- **Position** feature describes if the current word is before or after the predicate.
- **Chunk pattern** encodes the sequence of chunks from the current words to the predicate.
- **Clause tag** indicates the boundary of clauses.
- **Clause path** feature is a path formed from a semi-parsed tree containing only clauses and chunks. Each clause is named with the chunk preceding it. The clause path is the path from predicate to target word in the semi-parse tree.
- **Clause position** feature is the position of the target word relative to the predicate in the semi-parse tree containing only clauses. There are four configurations – target word and predicate share the same parent, target word parent is an ancestor of predicate, predicate parent is an ancestor of target word, or otherwise.

Because each argument consists of a single beginning and a single ending, these classifiers can be used to construct a set of potential arguments (by combining each predicted *begin* with each predicted *end* after it of the same type).

Although this phase identifies typed arguments (i.e. labeled with argument types), the second phase will re-score each phrase using phrase-based classifiers – therefore, the goal of the first phase is simply to identify non-typed phrase candidates. In this task, we achieves 98.96% and 88.65% recall (overall, without verb) on the training and the development set, respectively. Because these are the only candidates passed to the second phase, the final system performance is upper-bounded by 88.65%.

3.3 Second Phase: Argument Classification

The second phase of our system assigns the final argument classes to (a subset) of the argument candidates supplied from the first phase. Again, the SNoW learning architecture is used to train a multi-class classifier to label each argument to one of the argument types, plus a special class—*no argument (null)*. Training examples are created from the argument candidates supplied from the first phase using the following features:

- **Predicate lemma & POS tag, voice, position, clause Path, clause position, chunk pattern** Same features as those in the first phase.
- **Word & POS tag** from the argument, including the first,last,and head[1] word and tag.
- **Named entity** feature tells if the target argument is, embeds, overlaps, or is embedded in a named entity with its type.
- **Chunk** tells if the target argument is, embeds, overlaps, or is embedded in a chunk with its type.
- **Lengths** of the target argument, in the numbers of words and chunks separately.
- **Verb class** feature is the class of the active predicate described in PropBank Frames.
- **Phrase type** uses simple heuristics to identify the target argument as VP, PP, or NP.
- **Sub-categorization** describes the phrase structure around the predicate. We separate the clause where the predicate is in into three parts—the predicate chunk, segments before and after the predicate, and use the sequence of phrase types of these three segments.
- **Baseline** features identified *not* in the main verb chunk as AM-NEG and modal verb in the main verb chunk as AM-MOD.
- **Clause coverage** describes how much of the local clause (from the predicate) is covered by the target argument.
- **Chunk pattern length** feature counts the number of patterns in the argument.
- **Conjunctions** join every pair of the above features as new features.
- **Boundary words & POS tag** include two words/tags before and after the target argument.
- **Bigrams** are pairs of words/tags in the window from two words before the target to the first word of the target, and also from the last word to two words after the argument.

[1]We use simple rules to first decide if a candidate phrase type is VP, NP, or PP. The headword of an NP phrase is the right-most noun. Similarly, the left-most verb/proposition of a VP/PP phrase is extracted as the headword

- **Sparse collocation** picks one word/tag from the two words before the argument, the first word/tag, the last word/tag of the argument, and one word/tag from the two words after the argument to join as features.

Although the predictions of the second-phase classifier can be used directly, the labels of arguments in a sentence often violate some constraints. Therefore, we rely on the inference procedure to make the final predictions.

4 Inference via ILP

Ideally, if the learned classifiers are perfect, arguments can be labeled correctly according to the classifiers' predictions. In reality, labels assigned to arguments in a sentence often contradict each other, and violate the constraints arising from the structural and linguistic information. In order to resolve the conflicts, we design an inference procedure that takes the confidence scores of each individual argument given by the second-phase classifier as input, and outputs the *best* global assignment that also satisfies the constraints. In this section we first introduce the constraints and the inference problem in the semantic role labeling task. Then, we demonstrate how we apply integer linear programming(ILP) to reason for the global label assignment.

4.1 Constraints over Argument Labeling

Formally, the argument classifier attempts to assign labels to a set of arguments, $S^{1:M}$, indexed from 1 to M. Each argument S^i can take any label from a set of argument labels, $\mathcal{P}$, and the indexed set of arguments can take a set of labels, $c^{1:M} \in \mathcal{P}^M$. If we assume that the classifier returns a score, $\text{score}(S^i = c^i)$, corresponding to the likelihood of seeing label c^i for argument S^i, then, given a sentence, the unaltered inference task is solved by maximizing the overall score of the arguments,

$$\begin{aligned} \hat{c}^{1:M} &= \underset{c^{1:M} \in \mathcal{P}^M}{\text{argmax}} \, \text{score}(S^{1:M} = c^{1:M}) \\ &= \underset{c^{1:M} \in \mathcal{P}^M}{\text{argmax}} \sum_{i=1}^{M} \text{score}(S^i = c^i). \end{aligned} \quad (1)$$

In the presence of global constraints derived from linguistic information and structural considerations, our system seeks for a *legitimate* labeling that maximizes the score. Specifically, it can be viewed as the solution space is limited through the use of a filter function, $\mathcal{F}$, that eliminates many argument labelings from consideration. It is interesting to contrast this with previous work that filters individual phrases (see (Carreras and Màrquez, 2003)). Here, we are concerned with global constraints as well as constraints on the arguments. Therefore, the final labeling becomes

$$\hat{c}^{1:M} = \underset{c^{1:M} \in \mathcal{F}(\mathcal{P}^M)}{\text{argmax}} \sum_{i=1}^{M} \text{score}(S^i = c^i) \quad (2)$$

The filter function used considers the following constraints:

1. Arguments cannot cover the predicate except those that contain only the verb or the verb and the following word.
2. Arguments cannot overlap with the clauses (they can be embedded in one another).
3. If a predicate is outside a clause, its arguments cannot be embedded in that clause.
4. No overlapping or embedding arguments.
5. No duplicate argument classes for A0–A5,V.
6. Exactly one V argument per verb.
7. If there is C-V, then there should be a sequence of consecutive V, A1, and C-V pattern. For example, when *split* is the verb in "split it up", the A1 argument is "it" and C-V argument is "up".
8. If there is an R-XXX argument, then there has to be an XXX argument. That is, if an argument is a reference to some other argument XXX, then this referenced argument must exist in the sentence.
9. If there is a C-XXX argument, then there has to be an XXX argument; in addition, the C-XXX argument must occur after XXX. This is stricter than the previous rule because the order of appearance also needs to be considered.
10. Given the predicate, some argument classes are illegal (e.g. predicate 'stalk' can take only A0 or A1). This linguistic information can be found in *PropBank Frames*.

We reformulate the constraints as linear (in)equalities by introducing indicator variables. The optimization problem (Eq. 2) is solved using ILP.

4.2 Using Integer Linear Programming

As discussed previously, a collection of potential arguments is not necessarily a valid semantic labeling since it must satisfy all of the constraints. In this context, inference is the process of finding the *best* (according to Equation 1) valid semantic labels that satisfy all of the specified constraints. We take a similar approach that has been previously used

for entity/relation recognition (Roth and Yih, 2004), and model this inference procedure as solving an ILP.

An *integer linear program*(ILP) is basically the same as a *linear program*. The cost function and the (in)equality constraints are all linear in terms of the variables. The only difference in an ILP is the variables can only take integers as their values. In our inference problem, the variables are in fact binary. A general binary integer programming problem can be stated as follows.

Given a cost vector $\mathbf{p} \in \Re^d$, a set of variables, $\mathbf{z} = (z_1, \ldots, z_d)$ and cost matrices $\mathbf{C}_1 \in \Re^{t_1} \times \Re^d$, $\mathbf{C}_2 \in \Re^{t_2} \times \Re^d$, where t_1 and t_2 are the numbers of inequality and equality constraints and d is the number of binary variables. The ILP solution $\mathbf{z}^*$ is the vector that maximizes the cost function,

$$\mathbf{z}^* = \underset{\mathbf{z} \in \{0,1\}^d}{\operatorname{argmax}} \mathbf{p} \cdot \mathbf{z},$$

$$\text{subject to } \mathbf{C}_1\mathbf{z} \geq \mathbf{b}_1, \text{ and } \mathbf{C}_2\mathbf{z} = \mathbf{b}_2,$$

where $\mathbf{b}_1, \mathbf{b}_2 \in \Re^d$, and for all $z \in \mathbf{z}$, $z \in \{0, 1\}$.

To solve the problem of Equation 2 in this setting, we first reformulate the original cost function $\sum_{i=1}^{M} \text{score}(S^i = c^i)$ as a linear function over several binary variables, and then represent the filter function $\mathcal{F}$ using linear inequalities and equalities.

We set up a bijection from the semantic labeling to the variable set $\mathbf{z}$. This is done by setting $\mathbf{z}$ to a set of indicator variables. Specifically, let $z_{ic} = [S^i = c]$ be the indicator variable that represents whether or not the argument type c is assigned to S^i, and let $p_{ic} = \text{score}(S^i = c)$. Equation 1 can then be written as an ILP cost function as

$$\underset{\mathbf{z} \in \{0,1\}^d}{\operatorname{argmax}} \sum_{i=1}^{M} \sum_{c=1}^{|\mathcal{P}|} p_{ic} z_{ic},$$

subject to

$$\sum_{c=1}^{|\mathcal{P}|} z_{ic} = 1 \qquad \forall z_{ic} \in \mathbf{z},$$

which means that each argument can take only one type. Note that this new constraint comes from the variable transformation, and is not one of the constraints used in the filter function $\mathcal{F}$.

Constraints 1 through 3 can be evaluated on a per-argument basis – the sake of efficiency, arguments that violate these constraints are eliminated even before given the second-phase classifier. Next, we show how to transform the constraints in the filter function into the form of linear (in)equalities over $\mathbf{z}$, and use them in this ILP setting.

Constraint 4: No overlapping or embedding If arguments $S^{j_1}, \ldots, S^{j_k}$ occupy the same word in a sentence, then this constraint restricts only one arguments to be assigned to an argument type. In other words, $k - 1$ arguments will be the special class *null*, which means the argument candidate is not a legitimate argument. If the special class *null* is represented by the symbol ϕ, then for every set of such arguments, the following linear equality represents this constraint.

$$\sum_{i=1}^{k} z_{j_i\phi} = k - 1$$

Constraint 5: No duplicate argument classes Within the same sentence, several types of arguments cannot appear more than once. For example, a predicate can only take one A0. This constraint can be represented using the following inequality.

$$\sum_{i=1}^{M} z_{i\text{A0}} \leq 1$$

Constraint 6: Exactly one V argument For each verb, there is one and has to be one V argument, which represents the active verb. Similarly, this constraint can be represented by the following equality.

$$\sum_{i=1}^{M} z_{i\text{V}} = 1$$

Constraint 7: V–A1–C-V pattern This constraint is only useful when there are three consecutive candidate arguments in a sentence. Suppose arguments $S^{j_1}, S^{j_2}, S^{j_3}$ are consecutive. If S^{j_3} is C-V, then S^{j_1} and S^{j_2} have to be V and A1, respectively. This if-then constraint can be represented by the following two linear inequalities.

$$z_{j_3\text{C-V}} \geq z_{j_1\text{V}}, \text{ and } z_{j_3\text{C-V}} \geq z_{j_2\text{A1}}$$

Constraint 8: R-XXX arguments Suppose the referenced argument type is A0 and the reference type is R-A0. The linear inequalities that represent this constraint are:

$$\forall m \in \{1, \ldots, M\} : \sum_{i=1}^{M} z_{i\text{A0}} \geq z_{m\text{R-A0}}$$

If there are γ reference argument pairs, then the total number of inequalities needed is γM.

Constraint 9: C-XXX arguments This constraint is similar to the reference argument constraints. The difference is that the continued argument XXX has to occur before C-XXX. Assume that the argument pair is A0 and C-A0, and argument S_{j_i} appears before S_{j_k} if $i \leq k$. The linear inequalities that represent this constraint are:

$$\forall m \in \{2, \ldots, M\} : \sum_{i=1}^{j-1} z_{j_i\text{A0}} \geq z_{m\text{R-A0}}$$

Constraint 10: Illegal argument types Given a specific verb, some argument types should never occur. For example, most verbs don't have arguments A5. This constraint is represented by summing all the corresponding indicator variables to be 0.

$$\sum_{i=1}^{M} z_{i\text{A5}} = 0$$

Using ILP to solve this inference problem enjoys several advantages. Linear constraints are very general, and are able to represent many types of constraints. Previous approaches usually rely on dynamic programming to resolve non overlapping/embedding constraints (i.e., Constraint 4) when the data is sequential, but are unable to handle other constraints. The ILP approach is flexible enough to handle constraints regardless of the structure of the data. Although solving an ILP problem is NP-hard, with the help of todays commercial numerical packages, this problem can usually be solved very fast in practice. For instance, it only takes about 10 minutes to solve the inference problem for 4305 sentences on a Pentium-III 800 MHz machine in our experiments. Note that ordinary search methods (e.g., beam search) are not necessarily faster than solving an ILP problem and do not guarantee the optimal solution.

5 Experimental Results

The system is evaluated on the data provided in the CoNLL-2004 semantic-role labeling shared task which consists of a portion of PropBank corpus. The training set is extracted from TreeBank (Marcus et al., 1993) section 15–18, the development set, used in tuning parameters of the system, from section 20, and the test set from section 21.

We first compare this system with the basic tagger that we have, the CSCL shallow parser from (Punyakanok and Roth, 2001), which is equivalent to using the scoring function from the first phase with only the non-overlapping/embedding constraints. In addition, we evaluate the effectiveness of using only this constraint versus all constraints, as in Sec. 4.

	Prec.	Rec.	$F_{\beta=1}$
1st-phase, non-overlap	70.54	61.50	65.71
1st-phase, All Const.	70.97	60.74	65.46
2nd-phase, non-overlap	69.69	64.75	67.13
2nd-phase, All Const.	71.96	64.93	68.26

Table 1: Summary of experiments on the development set. All results are for overall performance.

	Precision	Recall	$F_{\beta=1}$
Without Inference	86.95	87.24	87.10
With Inference	88.03	88.23	88.13

Table 2: Results of second phase phrase prediction and inference assuming *perfect boundary detection* in the first phase. Inference improves performance by restricting label sequences rather than restricting structural properties since the correct boundaries are given. All results are for overall performance on the development set.

Table 1 shows how additional constraints over the standard non-overlapping constraints improve performance on the development set. The argument scoring is chosen from either the first phase or the second phase and each is evaluated by considering simply the non-overlapping/embedding constraint or the full set of linguistic constraints. To make a fair comparison, parameters were set separately to optimize performance when using the first phase results. In general, using all constraints increases $F_{\beta=1}$ by about 1% in this system, but slightly decreases the performance when only the first phase classifier is used. Also, using the two-phase architecture improves both precision and recall, and the enhancement reflected in $F_{\beta=1}$ is about 2.5%.

It is interesting to find out how well the second phase classifier can perform given perfectly segmented arguments. This evaluates the quality of the argument classifier, and also provides a conceptual upper bound. Table 2 first shows the results without using inference (i.e. $\mathcal{F}(\mathcal{P}^M) = \mathcal{P}^M$). The second row shows adding inference to the phrase classification can further improve $F_{\beta=1}$ by 1%.

Finally, the overall result on the official test set is given in Table 3. Note that the result here is not comparable with the best in this domain (Pradhan et al., 2004) where the full parse tree is assumed given. For a fair comparison, our system was among the best at CoNLL-04, where the best system (Hacioglu et al., 2004) achieve a 69.49 F1 score.

6 Conclusion

We show that linguistic information is useful for semantic role labeling, both in extracting features and

	Dist.	Prec.	Rec.	$F_{\beta=1}$
Overall	100.00	70.07	63.07	66.39
A0	26.87	81.13	77.70	79.38
A1	35.73	74.21	63.02	68.16
A2	7.44	54.16	41.04	46.69
A3	1.56	47.06	26.67	34.04
A4	0.52	71.43	60.00	65.22
AM-ADV	3.20	39.36	36.16	37.69
AM-CAU	0.51	45.95	34.69	39.53
AM-DIR	0.52	42.50	34.00	37.78
AM-DIS	2.22	52.00	67.14	58.61
AM-EXT	0.15	46.67	50.00	48.28
AM-LOC	2.38	33.47	34.65	34.05
AM-MNR	2.66	45.19	36.86	40.60
AM-MOD	3.51	92.49	94.96	93.70
AM-NEG	1.32	85.92	96.06	90.71
AM-PNC	0.89	32.79	23.53	27.40
AM-TMP	7.78	59.77	56.89	58.30
R-A0	1.66	81.33	76.73	78.96
R-A1	0.73	58.82	57.14	57.97
R-A2	0.09	100.00	22.22	36.36
R-AM-TMP	0.15	54.55	42.86	48.00

Table 3: Results on the test set.

deriving hard constraints on the output. We also demonstrate that it is possible to use integer linear programming to perform inference that incorporates a wide variety of hard constraints, which would be difficult to incorporate using existing methods. In addition, we provide further evidence supporting the use of scoring arguments over scoring argument boundaries for complex tasks. In the future, we plan to use the full PropBank corpus to see the improvement when more training data is provided. In addition, we would like to explore the possibility of integer linear programming approach using soft constraints. As more constraints are considered, we expect the overall performance to improve.

7 Acknowledgments

We thank Xavier Carreras and Lluís Màrquez for the data and scripts, Martha Palmer and the anonymous referees for their useful comments, AMD for their equipment donation, and Dash Optimization for the free academic use of their Xpress-MP software. This research is supported by NSF grants ITR-IIS-0085836, ITR-IIS-0085980 and IIS-9984168, EIA-0224453 and an ONR MURI Award.

References

C. Bishop, 1995. *Neural Networks for Pattern Recognition*, chapter 6.4: Modelling conditional distributions, page 215. Oxford University Press.

X. Carreras and L. Màrquez. 2003. Phrase recognition by filtering and ranking with perceptrons. In *Proc. of RANLP-2003*.

J. Chen and O. Rambow. 2003. Use of deep linguistic features for the recognition and labeling of semantic arguments. In *Proc. of EMNLP-2003*, Sapporo, Japan.

D. Gildea and J. Hockenmaier. 2003. Identifying semantic roles using combinatory categorial grammar. In *Proc. of the EMNLP-2003*, Sapporo, Japan.

D. Gildea and M. Palmer. 2002. The necessity of parsing for predicate argument recognition. In *Proc. of ACL 2002*, pages 239–246, Philadelphia, PA.

A. Grove and D. Roth. 2001. Linear concepts and hidden variables. *Machine Learning*, 42(1/2):123–141.

K. Hacioglu, S. Pradhan, W. Ward, J. H. Martin, and D. Jurafsky. 2004. Semantic role labeling by tagging syntactic chunks. In *Proc. of CoNLL-04*.

T. Hang, F. Damerau, and D. Johnson. 2002. Text chunking based on a generalization of winnow. *J. of Machine Learning Research*, 2:615–637.

P. Kingsbury and M. Palmer. 2002. From Treebank to PropBank. In *Proc. of LREC-2002*, Spain.

M. P. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, June.

S. Pradhan, K. Hacioglu, W. ward, J. Martin, and D. Jurafsky. 2003. Semantic role parsing adding semantic structure to unstructured text. In *Proc. of ICDM-2003*, Melbourne, FL.

S. Pradhan, W. Ward, K. Hacioglu, J. H. Martin, and D. Jurafsky. 2004. Shallow semantic parsing using support vector machines. In *Proc. of NAACL-HLT 2004*.

V. Punyakanok and D. Roth. 2001. The use of classifiers in sequential inference. In *NIPS-13; The 2000 Conference on Advances in Neural Information Processing Systems*, pages 995–1001. MIT Press.

D. Roth and W. Yih. 2002. Probabilistic reasoning for entity & relation recognition. In *Proc. of COLING-2002*, pages 835–841.

D. Roth and W. Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proc. of CoNLL-2004*.

D. Roth. 1998. Learning to resolve natural language ambiguities: A unified approach. In *Proc. of AAAI*, pages 806–813.

M. Surdeanu, S. Harabagiu, J. Williams, and P. Aarseth. 2003. Using predicate-argument structures for information extraction. In *Proc. of ACL 2003*.

E. F. Tjong Kim Sang and S. Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. In *Proc. of the CoNLL-2000 and LLL-2000*.

E. F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proc. of CoNLL-2003*.

E. F. Tjong Kim Sang and H. Déjean. 2001. Introduction to the CoNLL-2001 shared task: Clause identification. In *Proc. of the CoNLL-2001*.

Xpress-MP. 2003. Dash Optimization. Xpress-MP. http://www.dashoptimization.com/products.html.