# Semantic Role Labeling Via Generalized Inference Over Classifiers

**Vasin Punyakanok, Dan Roth, Wen-tau Yih, Dav Zimak**
Department of Computer Science

**Yuancheng Tu**
Department of Linguistics

University of Illinois at Urbana-Champaign
{punyakan,danr,yih,davzimak,ytu}@uiuc.edu

## Abstract

We present a system submitted to the CoNLL-2004 shared task for semantic role labeling. The system is composed of a set of classifiers and an inference procedure used both to *clean* the classification results and to ensure structural integrity of the final role labeling. Linguistic information is used to generate features during classification and constraints for the inference process.

## 1 Introduction

Semantic role labeling is a complex task to discover patterns within sentences corresponding to semantic meaning. We believe it is hopeless to 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. The system we present here is composed of two phases.

First, a set of phrase candidates is produced using two learned classifiers—one to discover beginning positions and one to discover end positions for each argument type. Hopefully, this phase discovers a small superset of all phrases in the sentence (for each verb).

In the second phase, the final prediction is made. First, candidate phrases from the first phase are re-scored using a classifier designed to determine argument type, given a candidate phrase. Because phrases are considered as a whole, global properties of the candidates can be used to discover how likely it is that a phrase is of a given argument type. However, the set of possible role-labelings is restricted by structural and linguistic constraints. We encode these constraints using linear functions and use integer programming to ensure the final prediction is consistent (see Section 4).

## 2 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 (phrase 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 update rule. In particular, a regularization term is added, which has the affect of trying to separate the data with a think 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. 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.

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

## 3 First Phase: Find Argument Candidates

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

Specifically, we learn two classifiers, one to detect beginning phrase locations and a second to detect end phrase locations. Each multi-class classifier makes predictions over forty-three classes – thirty-two argument types, ten continuous argument types, one class to detect *not begging* and one class to detect *not end*. The following features are used:

- **Word** feature includes the current word, two words before and two words after.
- **Part-of-speech tag** (POS) feature includes the POS tags of the current word, two words before and after.
- **Chunk** feature includes the BIO tags for chunks of the current word, two words before and after.
- **Predicate lemma & POS tag** show the lemma form and POS tag of the active predicate.
- **Voice** feature indicates 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 chuck and its POS tag is VBN(past participle) or it immediately follows a noun phrase.
- **Position** feature describes if the current word is before of after the predicate.
- **Chunk pattern** feature 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 immediately preceding it. The clause path is the path from predicate to target word in the semi-parsed tree.
- **Clause position** feature is the position of the target word relative to the predicate in the semi-parsed tree containing only clauses. Specifically, there are four configurations—target word and predicate share same parent, parent of target word is ancestor of predicate, parent of predicate is ancestor of target word, or otherwise.

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

Although the outputs of this phase are potential argument candidates, along with their types, the second phase re-scores the arguments using all possible types. After eliminating the types from consideration, the first phase achieves $98.96\%$ and $88.65\%$ recall (overall, without verb) on the training and the development set, respectively. Because these are the only candidates that are passed to the second phase, $88.65\%$ is an upper bound of the recall for our overall system.

## 4 Second Phase: Phrase Classification

The second phase of our system assigns the final argument classes to (a subset) of the phrases supplied from the first phase. This task is accomplished in two steps. First, a multi-class classifier is used to supply confidence scores corresponding to how likely individual phrases are to have specific argument types. Then we look for the most likely solution over the whole sentence, given the matrix of confidences and linguistic information that serves as a set of global constraints over the solution space.

Again, the SNoW learning architecture is used to train a multi-class classifier to label each phrase to one of the argument types, plus a special class – *no argument*. Training examples are created from the phrase 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 the first phase.
- **Word & POS tag** from the phrase, including the first/last word and tag, and the head word[1].
- **Named entity** feature tells if the target phrase is, embeds, overlaps, or is embedded in a named entity.
- **Chunk** features are the same as named entity (but with chunks, e.g. noun phrases).
- **Length** of the target phrase, in the numbers of words and chunks.
- **Verb class** feature is the class of the active predicate described in the frame files.
- **Phrase type** uses simple heuristics to identify the target phrase like VP, PP, or NP.
- **Sub-categorization** describes the phrase structure around the predicate. We separate the clause where the predicate is in into three part – the predicate chunk, segments before and after the predicate. The sequence of the phrase types of these three segments is our feature.
- **Baseline** follows the rule of identifying AM-NEG and AM-MOD and uses them as features.
- **Clause coverage** describes how much of local clause (from the predicate) is covered by the target phrase.
- **Chunk pattern length** feature counts the number of patterns in the phrase.
- **Conjunctions** join every pair of the above features as new features.
- **Boundary words & POS tags** include one or two words/tags before and after the target phrase.

[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

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

Alternately, we could have derived a scoring function from the first phase confidences of the open and closed predictors for each argument type. This method has proved useful in the literature for shallow parsing (Punyakanok and Roth, 2001). However, it is hoped that additional global features of the phrase would be necessary due to the variety and complexity of the argument types. See Table 1 for a comparison.

Formally (but very briefly), the phrase classifier is attempting to assign labels to a set of phrases, $S^{1:M}$, indexed from 1 to $M$. Each phrase $S^i$ can take any label from a set of phrase labels, $\mathcal{P}$, and the indexed set of phrases can take a set of labels, $s^{1:M} \in \mathcal{P}^M$. If we assume that the classifier returns a score, $\mathrm{score}(S^i = s^i)$, corresponding to the likelihood of seeing label $s^i$ for phrase $S^i$, then, given a sentence, the unaltered inference task that is solved by our system maximizes the score of the phrase, $\mathrm{score}(S^{1:M} = s^{1:M})$,

$$\begin{aligned}\hat{s}^{1:M} &= \underset{s^{1:M} \in \mathcal{P}^M}{\mathrm{argmax}}\ \mathrm{score}(S^{1:M} = s^{1:M}) \\ &= \underset{s^{1:M} \in \mathcal{P}^M}{\mathrm{argmax}} \sum_{i=1}^{M} \mathrm{score}(S^i = s^i).\end{aligned} \tag{1}$$

The second step for phrase identification is eliminating labelings using global constraints derived from linguistic information and structural considerations. Specifically, we limit the solution space through the used of a filter function, $\mathcal{F}$, that eliminates many phrase 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 phrases. Therefore, the final labeling becomes

$$\hat{s}^{1:M} = \underset{s^{1:M} \in \mathcal{F}(\mathcal{P}^M)}{\mathrm{argmax}} \sum_{i=1}^{M} \mathrm{score}(S^i = s^i) \tag{2}$$

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 phrases.
5. No duplicate argument classes for A0-A5,V.
6. Exactly one V argument per sentence.
7. If there is C-V, then there has to be a V-A1-CV pattern.
8. If there is a R-XXX argument, then there has to be a XXX argument.
9. If there is a C-XXX argument, then there has to be a XXX argument; in addition, the C-XXX argument must occur after XXX.
10. Given the predicate, some argument classes are illegal (e.g. predicate 'stalk' can take only A0 or A1).

Constraint 1 is valid because all the arguments of a predicate must lie outside the predicate. The exception is for the boundary of the predicate itself. Constraint 1 through constraint 3 are actually constraints that can be evaluated on a per-phrase basis and thus can be applied to the individual phrases at any time. For efficiency sake, we eliminate these even before the second phase scoring is begun. Constraints 5, 8, and 9 are valid for only a subset of the arguments.

These constraints are easy to transform into linear constraints (for example, for each class $c$, constraint 5 becomes $\sum_{i=1}^{M}[S^i = c] \leq 1$) [2]. Then the optimum solution of the cost function given in Equation 2 can be found by integer linear programming[3]. A similar method was used for entity/relation recognition (Roth and Yih, 2004).

Almost all previous work on shallow parsing and phrase classification has used Constraint 4 to ensure that there are no overlapping phrases. By considering additional constraints, we show improved performance (see Table 1).

## 5 Results

In this section, we present results. For the second phase, we evaluate the quality of the phrase predictor. The result first evaluates the phrase classifier, given the perfect phrase locations without using inference (i.e. $\mathcal{F}(\mathcal{P}^M) = \mathcal{P}^M$). The second, adds inference to the phrase classification over the perfect classifiers (see Table 2). We evaluate the overall performance of our system (without assuming perfect phrases) by training and evaluating the phrase classifier on the output from the first phase (see Table 3).

Finally,since this is a tagging task, we compare this system with the basic tagger that we have, the CLCL

[2]where $[x]$ is 1 if $x$ is true and 0 otherwise

[3](Xpress-MP, 2003) was used in all experiments to solve integer linear programming.

| | Precision | Recall | F1 |
|---|---|---|---|
| 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. The phrase scoring is choosen from either the first phase or the second phase and each is evaluated by considering simply non-overlapping constraints or the full set of linguistic constraints. To make a fair comparison, parameters were set seperately to optimize performance when using the first phase results. All results are for overall performance.

| | Precision | Recall | F1 |
|---|---|---|---|
| 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.

shallow parser from (Punyakanok and Roth, 2001), which is equivalent to using the scoring function from the first phase with only the non-overlapping constraints. Table 1 shows how how additional constraints over the standard non-overlapping constraints improve performance on the development set[4].

## 6 Conclusion

We show that linguistic information is useful for semantic role labeling used both to derive features and to derive hard constraints on the output. We show that it is possible to use integer linear programming to perform inference that incorporates a wide variety of hard constraints that would be difficult to incorporate using existing methods. In addition, we provide further evidence supporting the use of scoring phrases over scoring phrase boundaries for complex tasks.

**Acknowledgments** This research is supported by NSF grants ITR-IIS-0085836, ITR-IIS-0085980 and IIS-9984168, EIA-0224453 and an ONR MURI Award. We also thank AMD for their equipment donation and Dash Optimization for free academic use of their Xpress-MP software.

## 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 *Proceedings of RANLP-2003*.

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

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

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 *COLING 2002, The 19th International Conference on Computational Linguistics*, 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.

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

[4]The test set was not publicly available to evaluate these results.

| | Precision | Recall | $F_{\beta=1}$ |
|---|---|---|---|
| Overall | 70.07% | 63.07% | 66.39 |
| A0 | 81.13% | 77.70% | 79.38 |
| A1 | 74.21% | 63.02% | 68.16 |
| A2 | 54.16% | 41.04% | 46.69 |
| A3 | 47.06% | 26.67% | 34.04 |
| A4 | 71.43% | 60.00% | 65.22 |
| A5 | 0.00% | 0.00% | 0.00 |
| AM-ADV | 39.36% | 36.16% | 37.69 |
| AM-CAU | 45.95% | 34.69% | 39.53 |
| AM-DIR | 42.50% | 34.00% | 37.78 |
| AM-DIS | 52.00% | 67.14% | 58.61 |
| AM-EXT | 46.67% | 50.00% | 48.28 |
| AM-LOC | 33.47% | 34.65% | 34.05 |
| AM-MNR | 45.19% | 36.86% | 40.60 |
| AM-MOD | 92.49% | 94.96% | 93.70 |
| AM-NEG | 85.92% | 96.06% | 90.71 |
| AM-PNC | 32.79% | 23.53% | 27.40 |
| AM-PRD | 0.00% | 0.00% | 0.00 |
| AM-TMP | 59.77% | 56.89% | 58.30 |
| R-A0 | 81.33% | 76.73% | 78.96 |
| R-A1 | 58.82% | 57.14% | 57.97 |
| R-A2 | 100.00% | 22.22% | 36.36 |
| R-A3 | 0.00% | 0.00% | 0.00 |
| R-AM-LOC | 0.00% | 0.00% | 0.00 |
| R-AM-MNR | 0.00% | 0.00% | 0.00 |
| R-AM-PNC | 0.00% | 0.00% | 0.00 |
| R-AM-TMP | 54.55% | 42.86% | 48.00 |
| V | 98.37% | 98.37% | 98.37 |

Table 3: Results on the test set.