Generalized Inference with Multiple Semantic Role Labeling Systems

Peter Koomen **Vasin Punyakanok** **Dan Roth** **Wen-tau Yih**
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA
{pkoomen2,punyakan,danr,yih}@uiuc.edu

Abstract

We present an approach to semantic role labeling (SRL) that takes the output of multiple argument classifiers and combines them into a coherent predicate-argument output by solving an optimization problem. The optimization stage, which is solved via integer linear programming, takes into account both the recommendation of the classifiers and a set of problem specific constraints, and is thus used both to clean the classification results and to ensure structural integrity of the final role labeling. We illustrate a significant improvement in overall SRL performance through this inference.

1 SRL System Architecture

Our SRL system consists of four stages: *pruning*, *argument identification*, *argument classification*, and *inference*. In particular, the goal of pruning and argument identification is to identify argument candidates for a given verb predicate. The system only classifies the argument candidates into their types during the argument classification stage. Linguistic and structural constraints are incorporated in the inference stage to resolve inconsistent global predictions. The inference stage can take as its input the output of the argument classification of a single system or of multiple systems. We explain the inference for multiple systems in Sec. 2.

1.1 Pruning

Only the constituents in the parse tree are considered as argument candidates. In addition, our system exploits the heuristic introduced by (Xue and Palmer, 2004) to filter out very unlikely constituents. The heuristic is a recursive process starting from the verb whose arguments are to be identified. It first returns the siblings of the verb; then it moves to the parent of the verb, and collects the siblings again. The process goes on until it reaches the root. In addition, if a constituent is a `PP` (propositional phrase), its children are also collected. Candidates consisting of only a single punctuation mark are not considered.

This heuristic works well with the correct parse trees. However, one of the errors by automatic parsers is due to incorrect `PP` attachment leading to missing arguments. To attempt to fix this, we consider as arguments the combination of any consecutive `NP` and `PP`, and the split of `NP` and `PP` inside the `NP` that was chosen by the previous heuristics.

1.2 Argument Identification

The argument identification stage utilizes binary classification to identify whether a candidate is an argument or not. We train and apply the binary classifiers on the constituents supplied by the pruning stage. Most of the features used in our system are standard features, which include

- **Predicate and POS tag of predicate** indicate the lemma of the predicate and its POS tag.
- **Voice** indicates tbe voice of the predicate.
- **Phrase type** of the constituent.
- **Head word and POS tag of the head word** include head word and its POS tag of the constituent. We use rules introduced by (Collins, 1999) to extract this feature.
- **First and last words and POS tags** of the constituent.
- **Two POS tags before and after** the constituent.
- **Position** feature describes if the constituent is before or after the predicate relative to the position in the sentence.

- **Path** records the traversal path in the parse tree from the predicate to the constituent.
- **Subcategorization** feature describes the phrase structure around the predicate's parent. It records the immediate structure in the parse tree that expands to its parent.
- **Verb class** feature is the class of the active predicate described in PropBank Frames.
- **Lengths** of the target constituent, in the numbers of words and chunks separately.
- **Chunk** tells if the target argument is, embeds, overlaps, or is embedded in a chunk with its type.
- **Chunk pattern length** feature counts the number of chunks from the predicate to the argument.
- **Clause relative position** is the position of the target word relative to the predicate in the pseudo-parse tree constructed only from clause constituent. There are four configurations—target constituent and predicate share the same parent, target constituent parent is an ancestor of predicate, predicate parent is an ancestor of target word, or otherwise.
- **Clause coverage** describes how much of the local clause (from the predicate) is covered by the argument. It is round to the multiples of $1/4$.

1.3 Argument Classification

This stage assigns the final argument labels to the argument candidates supplied from the previous stage. A multi-class classifier is trained to classify the types of the arguments supplied by the argument identification stage. To reduce the excessive candidates mistakenly output by the previous stage, the classifier can also classify the argument as *NULL* ("not an argument") to discard the argument.

The features used here are the same as those used in the argument identification stage with the following additional features.

- **Syntactic frame** describes the sequential pattern of the noun phrases and the predicate in the sentence. This is the feature introduced by (Xue and Palmer, 2004).
- **Propositional phrase head** is the head of the first phrase after the preposition inside `PP`.
- **NEG and MOD** feature indicate if the argument is a baseline for AM-NEG or AM-MOD. The rules of the **NEG** and **MOD** features are used in a baseline SRL system developed by Erik Tjong Kim Sang (Carreras and Màrquez, 2004).
- **NE** indicates if the target argument is, embeds, overlaps, or is embedded in a named-entity along with its type.

1.4 Inference

The purpose of this stage is to incorporate some prior linguistic and structural knowledge, such as "arguments do not overlap" or "each verb takes at most one argument of each type." This knowledge is used to resolve any inconsistencies of argument classification in order to generate final legitimate predictions. We use the inference process introduced by (Punyakanok et al., 2004). The process is formulated as an integer linear programming (ILP) problem that takes as inputs the confidences over each type of the arguments supplied by the argument classifier. The output is the optimal solution that maximizes the linear sum of the confidence scores (e.g., the conditional probabilities estimated by the argument classifier), subject to the constraints that encode the domain knowledge.

Formally speaking, 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 argument classifier returns an estimated conditional probability distribution, $Prob(S^i = c^i)$, then, given a sentence, the inference procedure seeks an global assignment that maximizes the following objective function,

$$\hat{c}^{1:M} = \underset{c^{1:M} \in \mathcal{P}^M}{\operatorname{argmax}} \sum_{i=1}^{M} Prob(S^i = c^i),$$

subject to linguistic and structural constraints. In other words, this objective function reflects the expected number of correct argument predictions, subject to the constraints. The constraints are encoded as the followings.

- No overlapping or embedding arguments.
- No duplicate argument classes for A0-A5.
- Exactly one V argument per predicate considered.
- If there is C-V, then there has to be a V-A1-CV pattern.
- If there is an R-*arg* argument, then there has to be an *arg* argument.
- If there is a C-*arg* argument, there must be an *arg* argument; moreover, the C-*arg* argument must occur after *arg*.
- Given the predicate, some argument types are illegal (e.g. predicate 'stalk' can take only A0 or A1). The illegal types may consist of A0-A5 and their corresponding C-*arg* and R-*arg* arguments. For each predicate, we look for the minimum value of i such that the class Ai is mentioned in its frame file as well as its maximum value j. All argument types Ak such that $k < i$ or $k > j$ are considered illegal.

2 Inference with Multiple SRL Systems

The inference process allows a natural way to combine the outputs from multiple argument classifiers. Specifically, given k argument classifiers which perform classification on k argument sets, $\{S_1, \ldots, S_k\}$. The inference process aims to optimize the objective function:

$$\hat{c}^{1:N} = \underset{c^{1:N} \in \mathcal{P}^N}{\operatorname{argmax}} \sum_{i=1}^{N} Prob(S^i = c^i),$$

where $S^{1:N} = \bigcup_{i=1}^{k} S_i$, and

$$Prob(S^i = c^i) = \frac{1}{k} \sum_{j=1}^{k} Prob_j(S^i = c^i),$$

where $Prob_j$ is the probability output by system j.

Note that all systems may not output with the same set of argument candidates due to the pruning and argument identification. For the systems that do not output for any candidate, we assign the probability with a prior to this *phantom* candidate. In particular, the probability of the *NULL* class is set to be 0.6 based on empirical tests, and the probabilities of the other classes are set proportionally to their occurrence frequencies in the training data.

For example, Figure 1 shows the two candidate sets for a fragment of a sentence, "*..., traders say, unable to* ***cool*** *the selling panic in both stocks and futures.*" In this example, system A has two argument candidates, a_1 = "traders" and a_4 = "the selling panic in both stocks and futures"; system B has three argument candidates, b_1 = "traders", b_2 = "the selling panic", and b_3 = "in both stocks and futures". The phantom candidates are created for a_2, a_3, and b_4 of which probability is set to the prior.

Specifically for this implementation, we first train two SRL systems that use Collins' parser and Charniak's parser respectively. In fact, these two parsers have noticeably different output. In evaluation, we run the system that was trained with Charniak's parser 5 times with the top-5 parse trees output by Charniak's parser[1]. Together we have six different outputs per predicate. Per each parse tree output, we ran the first three stages, namely pruning, argument identification, and argument classification. Then a joint inference stage is used to resolve the inconsistency of the output of argument classification in these systems.

[1]The top parse tree were from the official output by CoNLL. The 2nd-5th parse trees were output by Charniak's parser.

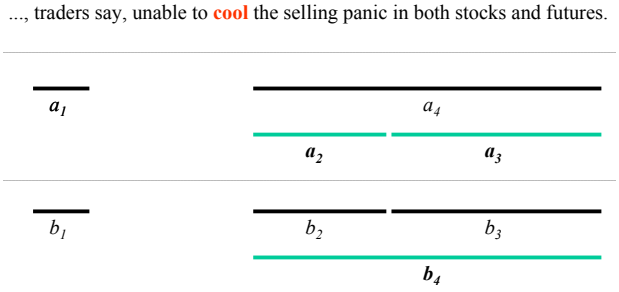

Figure 1: Two SRL systems' output (a_1, a_4, b_1, b_2, and b_3), and phantom candidates (a_2, a_3, and b_4).

3 Learning and Evaluation

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 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 improves the basic Winnow multiplicative update rule with a regularization term, which has the effect of trying to separate the data with a large margin separator (Grove and Roth, 2001; Hang et al., 2002) and voted (averaged) weight vector (Freund and Schapire, 1999).

Softmax function (Bishop, 1995) is used to convert raw activation to conditional probabilities. If there are n classes and the raw activation of class i is act_i, the posterior estimation for class i is

$$Prob(i) = \frac{e^{act_i}}{\sum_{1 \le j \le n} e^{act_j}}.$$

In summary, training used both full and partial syntactic information as described in Section 1. In training, SNoW's default parameters were used with the exception of the separator thickness 1.5, the use of average weight vector, and 5 training cycles. The parameters are optimized on the development set.

Training for each system took about 6 hours. The evaluation on both test sets which included running

	Precision	Recall	$F_{\beta=1}$
Development	80.05%	74.83%	77.35
Test WSJ	82.28%	76.78%	79.44
Test Brown	73.38%	62.93%	67.75
Test WSJ+Brown	81.18%	74.92%	77.92

Test WSJ	Precision	Recall	$F_{\beta=1}$
Overall	82.28%	76.78%	79.44
A0	88.22%	87.88%	88.05
A1	82.25%	77.69%	79.91
A2	78.27%	60.36%	68.16
A3	82.73%	52.60%	64.31
A4	83.91%	71.57%	77.25
A5	0.00%	0.00%	0.00
AM-ADV	63.82%	56.13%	59.73
AM-CAU	64.15%	46.58%	53.97
AM-DIR	57.89%	38.82%	46.48
AM-DIS	75.44%	80.62%	77.95
AM-EXT	68.18%	46.88%	55.56
AM-LOC	66.67%	55.10%	60.33
AM-MNR	66.79%	53.20%	59.22
AM-MOD	96.11%	98.73%	97.40
AM-NEG	97.40%	97.83%	97.61
AM-PNC	60.00%	36.52%	45.41
AM-PRD	0.00%	0.00%	0.00
AM-REC	0.00%	0.00%	0.00
AM-TMP	78.16%	76.72%	77.44
R-A0	89.72%	85.71%	87.67
R-A1	70.00%	76.28%	73.01
R-A2	85.71%	37.50%	52.17
R-A3	0.00%	0.00%	0.00
R-A4	0.00%	0.00%	0.00
R-AM-ADV	0.00%	0.00%	0.00
R-AM-CAU	0.00%	0.00%	0.00
R-AM-EXT	0.00%	0.00%	0.00
R-AM-LOC	85.71%	57.14%	68.57
R-AM-MNR	0.00%	0.00%	0.00
R-AM-TMP	72.34%	65.38%	68.69
V	98.92%	97.10%	98.00

Table 1: Overall results (top) and detailed results on the WSJ test (bottom).

with all six different parse trees (assumed already given) and the joint inference took about 4.5 hours.

	Precision	Recall	$F_{\beta=1}$
Charniak-1	75.40%	74.13%	74.76
Charniak-2	74.21%	73.06%	73.63
Charniak-3	73.52%	72.31%	72.91
Charniak-4	74.29%	72.92%	73.60
Charniak-5	72.57%	71.40%	71.98
Collins	73.89%	70.11%	71.95
Joint inference	80.05%	74.83%	77.35

Table 2: The results of individual systems and the result with joint inference on the development set.

Overall results on the development and test sets are shown in Table 1. Table 2 shows the results of individual systems and the improvement gained by the joint inference on the development set.

4 Conclusions

We present an implementation of SRL system which composed of four stages—1) pruning, 2) argument identification, 3) argument classification, and 4) inference. The inference provides a natural way to take the output of multiple argument classifiers and combines them into a coherent predicate-argument output. Significant improvement in overall SRL performance through this inference is illustrated.

Acknowledgments

We are grateful to Dash Optimization for the free academic use of Xpress-MP. This research is supported by ARDA's AQUAINT Program, DOI's Reflex program, 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. 2004. Introduction to the conll-2004 shared tasks: Semantic role labeling. In *Proc. of CoNLL-2004*.

M. Collins. 1999. *Head-driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, Computer Science Department, University of Pennsylvenia, Philadelphia.

Y. Freund and R. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296.

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, D. Roth, W. Yih, and D. Zimak. 2004. Semantic role labeling via integer linear programming inference. In *Proc. of COLING-2004*.

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

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

N. Xue and M. Palmer. 2004. Calibrating features for semantic role labeling. In *Proc. of the EMNLP-2004*, pages 88–94, Barcelona, Spain.