Semantic Role Labeling Using Complete Syntactic Analysis

Mihai Surdeanu
Technical University of Catalunya
surdeanu@lsi.upc.edu

Jordi Turmo
Technical University of Catalunya
turmo@lsi.upc.edu

Abstract

In this paper we introduce a semantic role labeling system constructed on top of the full syntactic analysis of text. The labeling problem is modeled using a rich set of lexical, syntactic, and semantic attributes and learned using one-versus-all AdaBoost classifiers.

Our results indicate that even a simple approach that assumes that each semantic argument maps into exactly one syntactic phrase obtains encouraging performance, surpassing the best system that uses partial syntax by almost 6%.

1 Introduction

Most current semantic role labeling (SRL) approaches can be classified in one of two classes: approaches that take advantage of complete syntactic analysis of text, pioneered by (Gildea and Jurafsky, 2002), and approaches that use partial syntactic analysis, championed by the previous CoNLL shared task evaluations (Carreras and Màrquez, 2004).

However, to the authors' knowledge, a clear analysis of the benefits of using full syntactic analysis versus partial analysis is not yet available. On one hand, the additional information provided by complete syntax should intuitively be useful. But, on the other hand, the state-of-the-art of full parsing is known to be less robust and perform worse than the tools used for partial syntactic analysis, which would decrease the quality of the information provided. The work presented in this paper contributes to this analysis by introducing a model that is entirely based on the full syntactic analysis of text, generated by a real-world parser.

2 System Description

2.1 Mapping Arguments to Syntactic Constituents

Our approach maps each argument label to one syntactic constituent, using a strategy similar to (Surdeanu et al., 2003). Using a bottom-up approach, we map each argument to the first phrase that has the exact same boundaries and climb as high as possible in the syntactic tree across unary production chains.

Unfortunately, this one-to-one mapping between semantic arguments and syntactic constituents is not always possible. One semantic argument may be mapped to many syntactic constituents due to: (a) intrinsic differences between the syntactic and semantic representations, and (b) incorrect syntactic structure. Figure 1 illustrates each one of these situations: Figure 1 (a) shows a sentence where each semantic argument correctly maps to one syntactic constituent; Figure 1 (b) illustrates the situation where one semantic argument correctly maps to two syntactic constituents; and Figure 1 (c) shows a one-to-many mapping caused by an incorrect syntactic structure: argument `A0` maps to two phrases, the terminal "by" and the noun phrase "Robert Goldberg", due to the incorrect attachment of the last prepositional phrase, "at the University of California".

Using the above observations, we separate one-

Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL),
pages 221–224, Ann Arbor, June 2005.

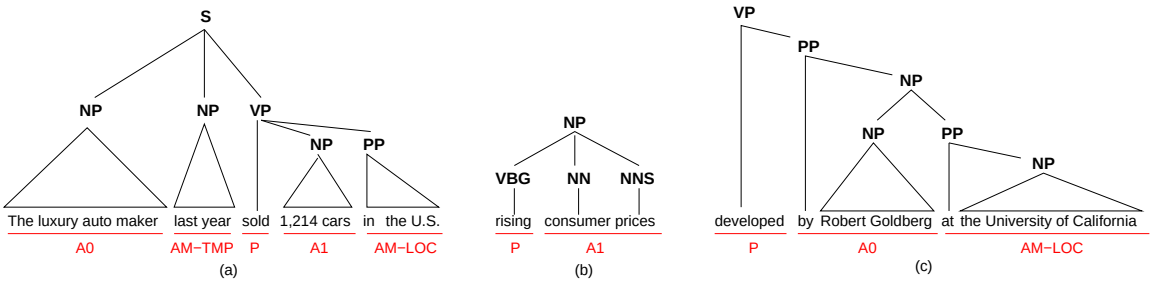

Figure 1: Mapping semantic arguments to syntactic constituents: (a) correct one–to-one mapping; (b) correct one-to-many mapping; (c) one-to-many mapping due to incorrect syntax.

	(a)	(b)	(c)
Training	96.06%	2.49%	1.45%
Development	91.36%	4.83%	3.81%

Table 1: Distribution of semantic arguments according to their mapping to syntactic constituents obtained with the Charniak parser: (a) one-to-one, (b) one-to-many, all syntactic constituents have same parent, (c) one-to-many, syntactic constituents have different parents.

to-many mappings in two classes: (a) when the syntactic constituents mapped to the semantic argument have the same parent (Figure 1 (b)) the mapping is correct and/or could theoretically be learned by a sequential SRL strategy, and (b) when the syntactic constituents mapped to the same argument have different parents, the mapping is generally caused by incorrect syntax. Such cases are very hard to be learned due to the irregularities of the parser errors.

Table 1 shows the distribution of semantic arguments into one of the above classes, using the syntactic trees provided by the Charniak parser. For the results reported in this paper, we model only one-to-one mappings between semantic arguments and syntactic constituents. A subset of the one-to-many mappings are addressed with a simple heuristic, described in Section 2.4.

2.2 Features

The features incorporated in the proposed model are inspired from the work of (Gildea and Jurafsky, 2002; Surdeanu et al., 2003; Pradhan et al., 2005; Collins, 1999) and can be classified into five classes: (a) features that capture the internal structure of the candidate argument, (b) features extracted from the argument context, (c) features that describe properties of the target predicate, (d) features generated from the predicate context, and (e) features that model the distance between the predicate and the argument. These five feature sets are listed in Tables 2, 3, 4, 5, and 6.

The *syntactic label* of the candidate constituent.
The constituent *head word*, *suffixes* of length 2, 3, and 4, *lemma*, and *POS tag*.
The constituent *content word*, *suffixes* of length 2, 3, and 4, *lemma*, *POS tag*, and *NE label*. Content words, which add informative lexicalized information different from the head word, were detected using the heuristics of (Surdeanu et al., 2003).
The *first and last constituent words* and their *POS tags*.
NE labels included in the candidate phrase.
Binary features to indicate the presence of *temporal cue words*, i.e. words that appear often in `AM-TMP` phrases in training.
For each TreeBank syntactic label we added a feature to indicate the *number of such labels* included in the candidate phrase.
The *sequence of syntactic labels* of the constituent immediate children.

Table 2: Argument structure features

The phrase *label, head word and POS tag* of the constituent parent, left sibling, and right sibling.

Table 3: Argument context features

2.3 Classifier

The classifiers used in this paper were developed using AdaBoost with confidence rated predictions (Schapire and Singer, 1999). AdaBoost combines many simple base classifiers or rules (in our case decision trees of depth 3) into a single strong classifier using a weighted-voted scheme. Each base classifier is learned sequentially from weighted examples and the weights are dynamically adjusted every learning iteration based on the behavior of the

The predicate *word* and *lemma*.
The predicate *voice*. We currently distinguish five voice types: active, passive, copulative, infinitive, and progressive.
A binary feature to indicate if the predicate is *frequent* - i.e. it appears more than twice in the training partition - or not.

Table 4: Predicate structure features

Sub-categorization rule, i.e. the phrase structure rule that expands the predicate immediate parent, e.g. NP → VBG NN NNS for the predicate in Figure 1 (b).

Table 5: Predicate context features

The *path* in the syntactic tree between the argument phrase and the predicate as a chain of syntactic labels along with the traversal direction (up or down).
The *length* of the above syntactic path.
The *number of clauses* (S* phrases) in the path.
The *number of verb phrases* (VP) in the path.
The *subsumption count*, i.e. the difference between the depths in the syntactic tree of the argument and predicate constituents. This value is 0 if the two phrases share the same parent.
The *governing category*, which indicates if NP arguments are dominated by a sentence (typical for subjects) or a verb phrase (typical for objects).
We *generalize* syntactic paths with more than 3 elements using two templates: (a) Arg ↑ Ancestor ↓ N_i ↓ Pred, where Arg is the argument label, Pred is the predicate label, Ancestor is the label of the common ancestor, and N_i is instantiated with all the labels between Pred and Ancestor in the full path; and (b) Arg ↑ N_i ↑ Ancestor ↓ Pred, where N_i is instantiated with all the labels between Arg and Ancestor in the full path.
The *surface distance* between the predicate and the argument phrases encoded as: the number of tokens, verb terminals (VB*), commas, and coordinations (CC) between the argument and predicate phrases, and a binary feature to indicate if the two constituents are adjacent.
A binary feature to indicate if the argument *starts with a predicate particle*, i.e. a token seen with the RP* POS tag and directly attached to the predicate in training.

Table 6: Predicate-argument distance features

previously learned rules.

We trained one-vs-all classifiers for the top 24 most common arguments in training (including R-A* and C-A*). For simplicity we do not label predicates. Following the strategy proposed by (Carreras et al., 2004) we select training examples (both positive and negative) only from: (a) the first S* phrase that includes the predicate, or (b) from phrases that appear to the left of the predicate in the sentence. More than 98% of the arguments fall into one of these classes.

At prediction time the classifiers are combined using a simple greedy technique that iteratively assigns to each predicate the argument classified with the highest confidence. For each predicate we consider as candidates all AM attributes, but only numbered attributes indicated in the corresponding PropBank frame.

2.4 Argument Expansion Heuristics

We address arguments that should map to more than one terminal phrase with the following post-processing heuristic: if an argument is mapped to one terminal phrase, its boundaries are extended to the right to include all terminal phrases that are not already labeled as other arguments for the same predicate. For example, after the system tags "consumer" as the beginning of an A1 argument in Figure 1, this heuristic extends the right boundary of the A1 argument to include the following terminal, "prices".

To handle inconsistencies in the treatment of quotes in parsing we added a second heuristic: arguments are expanded to include preceding/following quotes if the corresponding pairing quote is already included in the argument constituent.

3 Evaluation

3.1 Data

We trained our system using positive examples extracted from all training data available. Due to memory limitations on our development machines we used only the first 500,000 negative examples. In the experiments reported in this paper we used the syntactic trees generated by the Charniak parser. The results were evaluated for precision, recall, and F_1 using the scoring script provided by the task organizers.

3.2 Results and Discussion

Table 7 presents the results obtained by our system. On the WSJ data, our results surpass with almost 6% the results obtained by the best SRL system that used partial syntax in the CoNLL 2004 shared task evaluation (Hacioglu et al., 2004). Even though these numbers are not directly comparable (this year's shared task offers more training data), we consider these results encouraging given the simplicity of our system (we essentially model only one-to-one

	Precision	Recall	$F_{\beta=1}$
Development	79.14%	71.57%	75.17
Test WSJ	80.32%	72.95%	76.46
Test Brown	72.41%	59.67%	65.42
Test WSJ+Brown	79.35%	71.17%	75.04

Test WSJ	Precision	Recall	$F_{\beta=1}$
Overall	80.32%	72.95%	76.46
A0	87.09%	85.21%	86.14
A1	79.80%	72.23%	75.83
A2	74.74%	58.38%	65.55
A3	83.04%	53.76%	65.26
A4	77.42%	70.59%	73.85
A5	0.00%	0.00%	0.00
AM-ADV	57.82%	46.05%	51.27
AM-CAU	49.38%	54.79%	51.95
AM-DIR	62.96%	40.00%	48.92
AM-DIS	72.19%	76.25%	74.16
AM-EXT	60.87%	43.75%	50.91
AM-LOC	64.19%	52.34%	57.66
AM-MNR	63.90%	44.77%	52.65
AM-MOD	98.09%	93.28%	95.63
AM-NEG	96.15%	97.83%	96.98
AM-PNC	55.22%	32.17%	40.66
AM-PRD	0.00%	0.00%	0.00
AM-REC	0.00%	0.00%	0.00
AM-TMP	79.17%	73.41%	76.18
R-A0	84.85%	87.50%	86.15
R-A1	75.00%	71.15%	73.03
R-A2	60.00%	37.50%	46.15
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	68.00%	80.95%	73.91
R-AM-MNR	30.00%	50.00%	37.50
R-AM-TMP	60.81%	86.54%	71.43
V	0.00%	0.00%	0.00

Table 7: Overall results (top) and detailed results on the WSJ test (bottom).

mappings between semantic arguments and syntactic constituents). Only 0.14% out of the 75.17% F measure obtained on the development partition are attributed to the argument expansion heuristics introduced in Section 2.4.

4 Conclusions

This paper describes a semantic role labeling system constructed on top of the complete syntactic analysis of text. We model semantic arguments that map into exactly one syntactic phrase (about 90% of all semantic arguments in the development set) using a rich set of lexical, syntactic, and semantic attributes. We trained AdaBoost one-versus-all classifiers for the 24 most common argument types. Arguments that map to more than one syntactic constituent are expanded with a simple heuristic in a post-processing step.

Our results surpass with almost 6% the results obtained by best SRL system that used partial syntax in the CoNLL 2004 shared task evaluation. Although the two evaluations are not directly comparable due to differences in training set size, the current results are encouraging given the simplicity of our proposed system.

5 Acknowledgements

This research has been partially funded by the European Union project "Computers in the Human Interaction Loop" (CHIL - IP506909). Mihai Surdeanu is a research fellow within the Ramón y Cajal program of the Spanish Ministry of Education and Science.

We would also like to thank Lluís Màrquez and Xavi Carreras for the help with the AdaBoost classifier, for providing the set of temporal cue words, and for the many motivating discussions.

References

X. Carreras and L. Màrquez. 2004. Introduction to the CoNLL-2004 shared task: Semantic role labeling. In *Proceedings of CoNLL 2004 Shared Task*.

X. Carreras, L. Màrquez, and G. Chrupała. 2004. Hierarchical recognition of propositional arguments with perceptrons. In *Proceedings of CoNLL 2004 Shared Task*.

M. Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. PhD Dissertation, University of Pennsylvania.

D. Gildea and D. Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3).

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

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

R. E. Schapire and Y. Singer. 1999. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3).

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