Dependency-based Syntactic–Semantic Analysis with PropBank and NomBank

Richard Johansson and **Pierre Nugues**
Lund University, Sweden
{richard, pierre}@cs.lth.se

Abstract

This paper presents our contribution in the closed track of the 2008 CoNLL Shared Task (Surdeanu et al., 2008). To tackle the problem of joint syntactic–semantic analysis, the system relies on a syntactic and a semantic subcomponent. The syntactic model is a bottom-up projective parser using pseudo-projective transformations, and the semantic model uses global inference mechanisms on top of a pipeline of classifiers. The complete syntactic–semantic output is selected from a candidate pool generated by the subsystems.

The system achieved the top score in the closed challenge: a labeled syntactic accuracy of 89.32%, a labeled semantic F1 of 81.65, and a labeled macro F1 of 85.49.

1 Introduction: Syntactic–Semantic Analysis

Intuitively, semantic interpretation should help syntactic disambiguation, and joint syntactic–semantic analysis has a long tradition in linguistic theory. This motivates a statistical modeling of the problem of finding a syntactic tree $\hat{y}_{syn}$ and a semantic graph $\hat{y}_{sem}$ for a sentence $\boldsymbol{x}$ as maximizing a function F that scores the joint syntactic–semantic structure:

$$\langle \hat{y}_{syn}, \hat{y}_{sem} \rangle = \arg \max_{y_{syn}, y_{sem}} F(\boldsymbol{x}, y_{syn}, y_{sem})$$

The dependencies in the feature representation used to compute F determine the tractability of the search procedure needed to perform the maximization. To be able to use complex syntactic features such as paths when predicting semantic structures, exact search is clearly intractable. This is true even with simpler feature representations – the problem is a special case of multi-headed dependency analysis, which is NP-hard even if the number of heads is bounded (Chickering et al., 1994).

This means that we must resort to a simplification such as an incremental method or a reranking approach. We chose the latter option and thus created syntactic and semantic submodels. The joint syntactic–semantic prediction is selected from a small list of candidates generated by the respective subsystems.

2 Syntactic Submodel

We model the process of syntactic parsing of a sentence $\boldsymbol{x}$ as finding the parse tree $\hat{y}_{syn} = \arg\max_y F(\boldsymbol{x}, y)$ that maximizes a scoring function F. The learning problem consists of fitting this function so that the cost of the predictions is as low as possible according to a cost function ρ. In this work, we consider linear scoring functions of the following form:

$$F(\boldsymbol{x}, y) = \boldsymbol{w} \cdot \boldsymbol{\Psi}(\boldsymbol{x}, y)$$

where $\boldsymbol{\Psi}(\boldsymbol{x}, y)$ is a numeric feature representation of the pair $(\boldsymbol{x}, y)$ and $\boldsymbol{w}$ a vector of feature weights. We defined the syntactic cost ρ as the sum of link costs, where the link cost was 0 for a correct dependency link with a correct label, 0.5 for a correct link with an incorrect label, and 1 for an incorrect link.

A widely used framework for fitting the weight vector is the max-margin model (Taskar et al., 2003), which is a generalization of the well-known support vector machines to general cost-based prediction problems. Since the large number of training examples and features in our case make an exact solution of the max-margin optimization problem impractical, we used the online passive–aggressive algorithm (Crammer et al.,

CoNLL 2008: Proceedings of the 12th Conference on Computational Natural Language Learning, pages 183–187
Manchester, August 2008

2006), which approximates the optimization process in two ways:

- The weight vector $\boldsymbol{w}$ is updated incrementally, one example at a time.
- For each example, only the most violated constraint is considered.

The algorithm is a margin-based variant of the perceptron (preliminary experiments show that it outperforms the ordinary perceptron on this task). Algorithm 1 shows pseudocode for the algorithm.

Algorithm 1 The Online PA Algorithm

input Training set $\mathcal{T} = \{(\boldsymbol{x}_t, y_t)\}_{t=1}^{T}$
Number of iterations N
Regularization parameter C
Initialize w to zeros
repeat N times
  for $(\boldsymbol{x}_t, y_t)$ in $\mathcal{T}$
    let $\tilde{y}_t = \arg\max_y F(\boldsymbol{x}_t, y) + \rho(y_t, y)$
    let $\tau_t = \min\left(C, \frac{F(\boldsymbol{x}_t, \tilde{y}_t) - F(\boldsymbol{x}_t, y_t) + \rho(y_t, \tilde{y}_t)}{\|\boldsymbol{\Psi}(\boldsymbol{x}, y_t) - \boldsymbol{\Psi}(\boldsymbol{x}, \tilde{y}_t)\|^2}\right)$
    $\boldsymbol{w} \leftarrow \boldsymbol{w} + \tau_t(\boldsymbol{\Psi}(\boldsymbol{x}, y_t) - \boldsymbol{\Psi}(\boldsymbol{x}, \tilde{y}_t))$
return $\boldsymbol{w}_{\text{average}}$

We used a C value of 0.01, and the number of iterations was 6.

2.1 Features and Search

The feature function $\boldsymbol{\Psi}$ is a second-order edge-factored representation (McDonald and Pereira, 2006; Carreras, 2007). The second-order representation allows us to express features not only of head–dependent links, but also of siblings and children of the dependent. This feature set forces us to adopt the expensive search procedure by Carreras (2007), which extends Eisner's span-based dynamic programming algorithm (1996) to allow second-order feature dependencies. Since the cost function ρ is based on the cost of single links, this procedure can also be used to find the maximizer of $F(\boldsymbol{x}_i, y_{ij}) + \rho(y_i, y_{ij})$, which is needed at training time. The search was constrained to disallow multiple root links.

2.2 Handling Nonprojective Links

Although only 0.4% of the links in the training set are nonprojective, 7.6% of the sentences contain at least one nonprojective link. Many of these links represent long-range dependencies – such as *wh*-movement – that are valuable for semantic processing. Nonprojectivity cannot be handled by span-based dynamic programming algorithms. For parsers that consider features of single links only, the Chu-Liu/Edmonds algorithm can be used instead. However, this algorithm cannot be generalized to the second-order setting – McDonald and Pereira (2006) proved that this problem is NP-hard, and described an approximate greedy search algorithm.

To simplify implementation, we instead opted for the pseudo-projective approach (Nivre and Nilsson, 2005), in which nonprojective links are lifted upwards in the tree to achieve projectivity, and special trace labels are used to enable recovery of the nonprojective links at parse time. The use of trace labels in the pseudo-projective transformation leads to a proliferation of edge label types: from 69 to 234 in the training set, many of which occur only once. Since the running time of our parser depends on the number of labels, we used only the 20 most frequent trace labels.

3 Semantic Submodel

Our semantic model consists of three parts:

- A SRL classifier pipeline that generates a list of candidate predicate–argument structures.
- A constraint system that filters the candidate list to enforce linguistic restrictions on the global configuration of arguments.
- A global classifier that rescores the predicate–argument structures in the filtered candidate list.

Rather than training the models on gold-standard syntactic input, we created an automatically parsed training set by 5-fold cross-validation. Training on automatic syntax makes the semantic classifiers more resilient to parsing errors, in particular adjunct labeling errors.

3.1 SRL Pipeline

The SRL pipeline consists of classifiers for predicate identification, predicate disambiguation, support identification (for noun predicates), argument identification, and argument classification. We trained one set of classifiers for verb predicates and another for noun predicates. For the predicate disambiguation classifiers, we trained one subclassifier for each lemma. All classifiers in the pipeline were L2-regularized linear logistic regression classifiers, implemented using the efficient LIBLINEAR package (Lin et al., 2008). For multiclass problems, we used the one-vs-all binarization

method, which makes it easy to prevent outputs not allowed by the PropBank or NomBank frame.

Since our classifiers were logistic, their output values could be meaningfully interpreted as probabilities. This allowed us to combine the scores from subclassifiers into a score for the complete predicate–argument structure. To generate the candidate lists used by the global SRL models, we applied beam search based on these scores using a beam width of 4.

The features used by the classifiers are listed in Tables 1 and 2. In the tables, the features used by the classifiers for noun and verb predicates are indicated by N and V, respectively. We selected the feature sets by greedy forward subset selection.

Feature	PredId	PredDis
PREDWORD	N,V	N,V
PREDLEMMA	N,V	N,V
PREDPARENTWORD/POS	N,V	N,V
CHILDDEPSET	N,V	N,V
CHILDWORDSET	N,V	N,V
CHILDWORDDEPSET	N,V	N,V
CHILDPOSSET	N,V	N,V
CHILDPOSDEPSET	N,V	N,V
DEPSUBCAT	N,V	N,V
PREDRELTOPARENT	N,V	N,V

Table 1: Classifier features in predicate identification and disambiguation.

Feature	Supp	ArgId	ArgCl
PREDPARENTWORD/POS	N	N,V	
CHILDDEPSET	N	N,V	N,V
PREDLEMMASENSE	N	N,V	N,V
VOICE		V	V
POSITION	N	N,V	N,V
ARGWORD/POS	N	N,V	N,V
LEFTWORD/POS		N	N,V
RIGHTWORD/POS	N	N,V	N,V
LEFTSIBLINGWORD/POS			N,V
RIGHTSIBLINGWORD/POS		N	N
PREDPOS	N	N,V	V
RELPATH	N	N,V	N,V
POSPATH		N	
RELPATHTOSUPPORT		N	N
VERBCHAINHASSUBJ		V	V
CONTROLLERHASOBJ		V	N
PREDRELTOPARENT	N	N,V	N,V
FUNCTION			N,V

Table 2: Classifier features in argument identification and classification and support detection.

Features Used in Predicate Identification and Disambiguation

PREDWORD, PREDLEMMA. The lexical form and lemma of the predicate.

PREDPARENTWORD and PREDPARENTPOS. Form and part-of-speech tag of the parent node of the predicate.

CHILDDEPSET, CHILDWORDSET, CHILDWORDDEPSET, CHILDPOSSET, CHILDPOSDEPSET. These features represent the set of dependents of the predicate using combinations of dependency labels, words, and parts of speech.

DEPSUBCAT. Subcategorization frame: the concatenation of the dependency labels of the predicate dependents.

PREDRELTOPARENT. Dependency relation between the predicate and its parent.

Features Used in Argument Identification and Classification

PREDLEMMASENSE. The lemma and sense number of the predicate, e.g. *give.01*.

VOICE. For verbs, this feature is Active or Passive. For nouns, it is not defined.

POSITION. Position of the argument with respect to the predicate: Before, After, or On.

ARGWORD and ARGPOS. Lexical form and part-of-speech tag of the argument node.

LEFTWORD, LEFTPOS, RIGHTWORD, RIGHTPOS. Form/part-of-speech tag of the leftmost/rightmost dependent of the argument.

LEFTSIBLINGWORD, LEFTSIBLINGPOS, RIGHTSIBLINGWORD, RIGHTSIBLINGPOS. Form/part-of-speech tag of the left/right sibling of the argument.

PREDPOS. Part-of-speech tag of the predicate.

RELPATH. A representation of the complex grammatical relation between the predicate and the argument. It consists of the sequence of dependency relation labels and link directions in the path between predicate and argument, e.g. IM↑OPRD↑OBJ↓.

POSPATH. An alternative view of the grammatical relation, which consists of the POS tags passed when moving from predicate to argument, e.g. VB↑TO↑VBP↓PRP.

RELPATHTOSUPPORT. The RELPATH from the argument to a support chain.

VERBCHAINHASSUBJ. Binary feature that is set to true if the predicate verb chain has a subject. The purpose of this feature is to resolve verb coordination ambiguity as in Figure 1.

CONTROLLERHASOBJ. Binary feature that is true if the link between the predicate verb chain and its parent is OPRD, and the parent has an object. This feature is meant to resolve control ambiguity as in Figure 2.

FUNCTION. The grammatical function of the argument node. For direct dependents of the predicate, this is identical to the RELPATH.

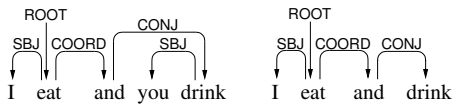

Figure 1: Coordination ambiguity: The subject *I* is in an ambiguous position with respect to *drink*.

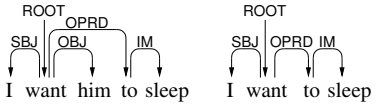

Figure 2: Subject/object control ambiguity: *I* is in an ambiguous position with respect to *sleep*.

3.2 Linguistically Motivated Global Constraints

The following three global constraints were used to filter the candidates generated by the pipeline.

CORE ARGUMENT CONSISTENCY. Core argument labels must not appear more than once.

DISCONTINUITY CONSISTENCY. If there is a label C-X, it must be preceded by a label X.

REFERENCE CONSISTENCY. If there is a label R-X and the label is inside a relative clause, it must be preceded by a label X.

3.3 Global SRL Model

Toutanova et al. (2005) have showed that a global model that scores the complete predicate–argument structure can lead to substantial performance gains. We therefore created a global SRL classifier using the following global features in addition to the features from the pipeline:

CORE ARGUMENT LABEL SEQUENCE. The complete sequence of core argument labels. The sequence also includes the predicate and voice, for instance `A0`+*break.01*/Active+`A1`.

MISSING CORE ARGUMENT LABELS. The set of core argument labels declared in the PropBank/NomBank frame that are not present in the predicate–argument structure.

Similarly to the syntactic submodel, we trained the global SRL model using the online passive–aggressive algorithm. The cost function ρ was defined as the number of incorrect links in the predicate–argument structure. The number of iterations was 20 and the regularization parameter C was 0.01. Interestingly, we noted that the global SRL model outperformed the pipeline even when no global features were added. This shows that the global learning model can correct label bias problems introduced by the pipeline architecture.

4 Syntactic–Semantic Integration

Our baseline joint feature representation contained only three features: the log probability of the syntactic tree and the log probability of the semantic structure according to the pipeline and the global model, respectively. This model was trained on the complete training set using cross-validation. The probabilities were obtained using the multinomial logistic function ("softmax").

We carried out an initial experiment with a more complex joint feature representation, but failed to improve over the baseline. Time prevented us from exploring this direction conclusively.

5 Results

The submitted results on the development and test corpora are presented in the upper part of Table 3. After the submission deadline, we corrected a bug in the predicate identification method. This resulted in improved results shown in the lower part.

Corpus	Syn acc	Sem F1	Macro F1
Development	88.47	80.80	84.66
Test WSJ	90.13	81.75	85.95
Test Brown	82.81	69.06	75.95
Test WSJ + Brown	89.32	80.37	84.86
Development	88.47	81.86	85.17
Test WSJ	90.13	83.75	86.61
Test Brown	82.84	69.85	76.34
Test WSJ + Brown	89.32	81.65	85.49

Table 3: Results.

5.1 Syntactic Results

Table 4 shows the effect of adding second-order features to the parser in terms of accuracy as well as training and parsing time on a Mac Pro, 3.2 GHz. The training times were measured on the complete training set and the parsing time and accuracies on the development set. Similarly to Carreras (2007), we see that these features have a very large impact on parsing accuracy, but also that the parser pays dearly in terms of efficiency as the search complexity increases from $O(n^3)$ to $O(n^4)$.

Since the low efficiency of the second-order parser restricts its use to batch applications, we see an interesting research direction to find suitable compromises between the two approaches, for instance by sacrificing the exact search procedure.

System	Training	Parse	Labeled	Unlabeled
1st order	65 min	28 sec	85.78	89.51
2nd order	60 hours	34 min	88.33	91.43

Table 4: Impact of second-order features.

Table 5 shows the dependency types most affected by the addition of second-order features to the parser when ordered by the increase in F1. As can be seen, they are all verb adjunct categories, which demonstrates the effect of grandchild features on PP attachment and labeling.

Label	ΔR	ΔP	ΔF_1
TMP	14.7	12.9	13.9
DTV	0	19.9	10.5
LOC	7.8	12.3	9.9
PRP	12.4	6.7	9.6
DIR	5.9	7.2	6.5

Table 5: Labels affected by second-order features.

5.2 Semantic Results

To assess the effect of the components in the semantic submodel, we tested their performance on the top-scoring parses from the syntactic model. Table 6 shows the results. The baseline system consists of the SRL pipeline only (P). Adding linguistic constraints (C) results in a more precision-oriented system with slightly lower recall, but significantly higher F1. Even higher performance is obtained when adding the global SRL model (G).

System	P	R	F1
P	80.74	77.98	79.33
P+C	82.42	77.66	79.97
P+C+G	83.64	78.14	80.40

Table 6: SRL results on the top-scoring parse trees.

5.3 Syntactic–Semantic Integration

The final experiment concerned the integration of syntactic and semantic analysis. In this setting, the system chooses the output that maximizes the joint syntactic–semantic score, based on the top N syntactic trees. Table 7 shows the results on the development set. We see that syntactic–semantic integration improves both syntactic accuracy and semantic F1. This holds for the constraint-based SRL system as well as for the full system.

Sem model	N	Syn acc	Sem F1	Macro F1
P+C	1	88.33	79.97	84.17
P+C	16	88.42	80.42	84.44
P+C+G	1	88.33	80.40	84.39
P+C+G	16	88.47	80.80	84.66

Table 7: Syntactic–semantic integration.

6 Conclusion

We have described a system[1] for syntactic and semantic dependency analysis based on PropBank and NomBank, and detailed the implementation of its subsystems. Crucial to our success was the high performance of the syntactic parser, which achieved a high accuracy. In addition, we reconfirmed the benefits of global inference in semantic analysis: both constraint-based and learning-based methods resulted in improvements over a baseline. Finally, we showed that integration of syntactic and semantic analysis is beneficial for both subtasks. We hope that this shared task will spur further research that leads to new feature representations and search procedures to handle the problem of joint syntactic and semantic analysis.

References

Carreras, Xavier. 2007. Experiments with a higher-order projective dependency parser. In *Proceedings of CoNLL*.

Chickering, David M., Dan Geiger, and David Heckerman. 1994. Learning Bayesian networks: The combination of knowledge and statistical data. Technical Report MSR-TR-94-09, Microsoft Research.

Crammer, Koby, Ofer Dekel, Joseph Keshet, Shai Shalev-Schwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *JMLR*, 2006(7):551–585.

Eisner, Jason M. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proc. of ICCL*.

Lin, Chih-Jen, Ruby C. Weng, and S. Sathiya Keerthi. 2008. Trust region Newton method for large-scale logistic regression. *JMLR*, 2008(9):627–650.

McDonald, Ryan and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL-2006*.

Nivre, Joakim and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proceedings of ACL-2005*.

Surdeanu, Mihai, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL–2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of CoNLL–2008*.

Taskar, Ben, Carlos Guestrin, and Daphne Koller. 2003. Max-margin Markov networks. In *Proceedings of NIPS*.

Toutanova, Kristina, Aria Haghighi, and Christopher D. Manning. 2005. Joint learning improves semantic role labeling. In *Proceedings of ACL-2005*.

[1] Our system is freely available for download at `http://nlp.cs.lth.se/lth_srl`.