Dependency-based Semantic Role Labeling of PropBank

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

Abstract

We present a PropBank semantic role labeling system for English that is integrated with a dependency parser. To tackle the problem of joint syntactic–semantic analysis, the system relies on a syntactic and a semantic subcomponent. The syntactic model is a 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.

We evaluate the system on the CoNLL-2005 test sets using segment-based and dependency-based metrics. Using the segment-based CoNLL-2005 metric, our system achieves a near state-of-the-art F1 figure of 77.97 on the WSJ+Brown test set, or 78.84 if punctuation is treated consistently. Using a dependency-based metric, the F1 figure of our system is 84.29 on the test set from CoNLL-2008. Our system is the first dependency-based semantic role labeler for PropBank that rivals constituent-based systems in terms of performance.

1 Introduction

Automatic semantic role labeling (SRL), the task of determining *who* does *what* to *whom*, is a useful intermediate step in NLP applications performing semantic analysis. It has obvious applications for template-filling tasks such as information extraction and question answering (Surdeanu et al., 2003; Moschitti et al., 2003). It has also been used in prototypes of NLP systems that carry out complex reasoning, such as entailment recognition systems (Haghighi et al., 2005; Hickl et al., 2006). In addition, role-semantic features have recently been used to extend vector-space representations in automatic document categorization (Persson et al., 2008).

The NLP community has recently devoted much attention to developing accurate and robust methods for performing role-semantic analysis automatically, and a number of multi-system evaluations have been carried out (Litkowski, 2004; Carreras and Màrquez, 2005; Baker et al., 2007; Surdeanu et al., 2008). Following the seminal work of Gildea and Jurafsky (2002), there have been many extensions in machine learning models, feature engineering (Xue and Palmer, 2004), and inference procedures (Toutanova et al., 2005; Surdeanu et al., 2007; Punyakanok et al., 2008).

With very few exceptions (e.g. Collobert and Weston, 2007), published SRL methods have used some sort of syntactic structure as input (Gildea and Palmer, 2002; Punyakanok et al., 2008). Most systems for automatic role-semantic analysis have used constituent syntax as in the Penn Treebank (Marcus et al., 1993), although there has also been much research on the use of shallow syntax (Carreras and Màrquez, 2004) in SRL.

In comparison, dependency syntax has received relatively little attention for the SRL task, despite the fact that dependency structures offer a more transparent encoding of predicate–argument relations. Furthermore, the few systems based on dependencies that have been presented have generally performed much worse than their constituent-based

Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing, pages 69–78,
Honolulu, October 2008.

counterparts. For instance, Pradhan et al. (2005) reported that a system using a rule-based dependency parser achieved much inferior results compared to a system using a state-of-the-art statistical constituent parser: The F-measure on WSJ section 23 dropped from 78.8 to 47.2, or from 83.7 to 61.7 when using a head-based evaluation. In a similar vein, Swanson and Gordon (2006) reported that parse tree path features extracted from a rule-based dependency parser are much less reliable than those from a modern constituent parser.

In contrast, we recently carried out a detailed comparison (Johansson and Nugues, 2008b) between constituent-based and dependency-based SRL systems for FrameNet, in which the results of the two types of systems where almost equivalent when using modern statistical dependency parsers. We suggested that the previous lack of progress in dependency-based SRL was due to low parsing accuracy. The experiments showed that the grammatical function information available in dependency representations results in a steeper learning curve when training semantic role classifiers, and it also seemed that the dependency-based role classifiers were more resilient to lexical problems caused by change of domain.

The recent CoNLL-2008 Shared Task (Surdeanu et al., 2008) was an attempt to show that SRL can be accurately carried out using only dependency syntax. However, these results are not easy to compare to previously published results since the task definitions and evaluation metrics were different.

This paper compares the best-performing system in the CoNLL-2008 Shared Task (Johansson and Nugues, 2008a) with previously published constituent-based SRL systems. The system carries out joint dependency-syntactic and semantic analysis. We first describe its implementation in Section 2, and then compare the system with the best system in the CoNLL-2005 Shared Task in Section 3. Since the outputs of the two systems are different, we carry out two types of evaluations: first by using the traditional segment-based metric used in the CoNLL-2005 Shared Task, and then by using the dependency-based metric from the CoNLL-2008 Shared Task. Both evaluations require a transformation of the output of one system: For the segment-based metric, we have to convert the dependency-based output to segments; and for the dependency-based metric, a head-finding procedure is needed to select heads in segments. For the first time for a system using only dependency syntax, we report results for PropBank-based semantic role labeling of English that are close to the state of the art, and for some measures even superior.

2 Syntactic–Semantic Dependency Analysis

The training corpus that we used is the dependency-annotated Penn Treebank from the 2008 CoNLL Shared Task on joint syntactic–semantic analysis (Surdeanu et al., 2008). Figure 1 shows a sentence annotated in this framework. The CoNLL task involved semantic analysis of predicates from PropBank (for verbs, such as *plan*) and NomBank (for nouns, such as *investment*); in this paper, we report the performance on PropBank predicates only since we compare our system with previously published PropBank-based SRL systems.

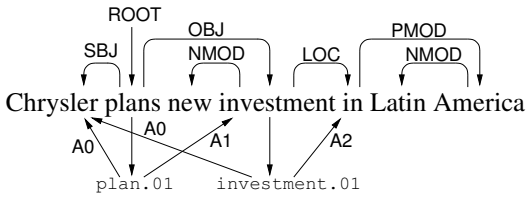

Figure 1: An example sentence annotated with syntactic and semantic dependency structures.

We model the problem of constructing a syntactic and a semantic graph as a task to be solved jointly. Intuitively, syntax and semantics are highly interdependent and semantic interpretation should help syntactic disambiguation, and joint syntactic–semantic analysis has a long tradition in deep-linguistic formalisms. Using a discriminative model, we thus formulate 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_{joint} that scores the complete syntactic–semantic structure:

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

The dependencies in the feature representation used to compute F_{joint} determine the tractability of the

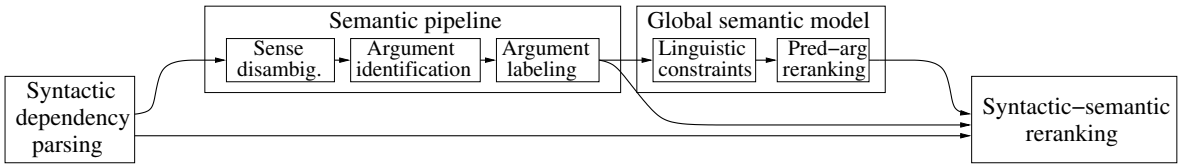

Figure 2: The architecture of the syntactic–semantic analyzer.

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. Figure 2 shows the architecture.

2.1 Syntactic Submodel

We model the process of syntactic parsing of a sentence $\boldsymbol{x}$ as finding the parse tree $\hat{y}_{syn} = \arg\max_{y_{syn}} F_{syn}(\boldsymbol{x}, y_{syn})$ that maximizes a scoring function F_{syn}. 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_{syn}. In this work, we consider linear scoring functions of the following form:

$$F_{syn}(\boldsymbol{x}, y_{syn}) = \boldsymbol{\Psi}_{syn}(\boldsymbol{x}, y_{syn}) \cdot \boldsymbol{w}$$

where $\boldsymbol{\Psi}_{syn}(\boldsymbol{x}, y_{syn})$ is a numeric feature representation of the pair $(\boldsymbol{x}, y_{syn})$ and $\boldsymbol{w}$ a vector of feature weights. We defined the syntactic cost ρ_{syn} 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 discriminative 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., 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.

2.1.1 Features and Search

The feature function $\boldsymbol{\Psi}_{syn}$ is a factored representation, meaning that we compute the score of the complete parse tree by summing the scores of its parts, referred to as *factors*:

$$\boldsymbol{\Psi}(\boldsymbol{x}, y) \cdot \boldsymbol{w} = \sum_{f \in y} \psi(\boldsymbol{x}, f) \cdot \boldsymbol{w}$$

We used a second-order factorization (McDonald and Pereira, 2006; Carreras, 2007), meaning that the factors are subtrees consisting of four links: the governor–dependent link, its sibling link, and the leftmost and rightmost dependent links of the dependent.

This factorization allows us to express useful features, but also 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. This algorithm has a time complexity of $O(n^4)$, where n is the number of words in the sentence. The search was constrained to disallow multiple root links.

To evaluate the $\arg\max$ in Algorithm 1 during training, we need to handle the cost function ρ_{syn} in addition to the factor scores. Since the cost function ρ_{syn} is based on the cost of single links, this can easily be integrated into the factor-based search.

2.1.2 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.

2.2 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 reranker that assigns scores to 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.

2.2.1 SRL Pipeline

The SRL pipeline consists of classifiers for predicate disambiguation, argument identification, and argument labeling. 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 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 argument identification classifier was preceded by a pruning step similar to the constituent-based pruning by Xue and Palmer (2004).

The features used by the classifiers are listed in Table 1, and are described in Appendix A. We selected the feature sets by greedy forward subset selection.

Feature	PredDis	ArgId	ArgLab
PREDWORD	•		
PREDLEMMA	•		
PREDPARENTWORD/POS	•		
CHILDDEPSET	•	•	•
CHILDWORDSET	•		
CHILDWORDDEPSET	•		
CHILDPOSSET	•		
CHILDPOSDEPSET	•		
DEPSUBCAT	•		
PREDRELTOPARENT	•		
PREDPARENTWORD/POS	•		
PREDLEMMASENSE		•	•
VOICE		•	•
POSITION		•	•
ARGWORD/POS		•	•
LEFTWORD/POS			•
RIGHTWORD/POS		•	•
LEFTSIBLINGWORD/POS			•
PREDPOS		•	•
RELPATH		•	•
VERBCHAINHASSUBJ		•	•
CONTROLLERHASOBJ		•	
PREDRELTOPARENT		•	•
FUNCTION			•

Table 1: Classifier features in predicate disambiguation (PredDis), argument identification (ArgId), and argument labeling (ArgLab).

2.2.2 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 an attributive relative clause, it must be preceded by a label X.

2.2.3 Predicate–Argument Reranker

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 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.

2.3 Syntactic–Semantic Reranking

As described previously, we carried out reranking on the candidate set of complete syntactic–semantic structures. To do this, we used the top 16 trees from the syntactic module and applied a linear model:

$$F_{joint}(\boldsymbol{x}, y_{syn}, y_{sem}) = \boldsymbol{\Psi}_{joint}(\boldsymbol{x}, y_{syn}, y_{sem}) \cdot \boldsymbol{w}$$

Our baseline joint feature representation $\boldsymbol{\Psi}_{joint}$ 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.

3 Comparisons with Previous Results

To compare our results with previously published results in SRL, we carried out an experiment comparing our system to the top system (Punyakanok et al., 2008) in the CoNLL-2005 Shared Task. However, comparison is nontrivial since the output of the CoNLL-2005 systems was a set of labeled segments, while the CoNLL-2008 systems (including ours) produced labeled semantic dependency links.

To have a fair comparison of our link-based system against previous segment-based systems, we

carried out a two-way evaluation: In the first evaluation, the dependency-based output was converted to segments and evaluated using the segment scorer from CoNLL-2005, and in the second evaluation, we applied a head-finding procedure to the output of a segment-based system and scored the result using the link-based CoNLL-2008 scorer.

It can be discussed which of the two metrics is most correlated with application performance. The traditional metric used in the CoNLL-2005 task treats SRL as a *bracketing problem*, meaning that the entities scored by the evaluation procedure are labeled snippets of text; however, it is questionable whether this is the proper way to evaluate a task whose purpose is to find *semantic relations* between logical entities. We believe that the same criticisms that have been leveled at the PARSEVAL metric for constituent structures are equally valid for the bracket-based evaluation of SRL systems. The inappropriateness of the traditional metric has led to a number of alternative metrics (Litkowski, 2004; Baker et al., 2007; Surdeanu et al., 2008).

3.1 Segment-based Evaluation

To be able to score the output of a dependency-based SRL system using the segment scorer, a conversion step is needed. Algorithm 2 shows how a set of segments is constructed from an argument dependency node. For each argument node, the algorithm computes the yield Y of the argument node, i.e. the set of dependency nodes to include in the bracketing. This set is then partitioned into contiguous parts, from which the segments are computed. In most cases, the yield is just the subtree dominated by the argument node. However, if the argument dominates the predicate, then the branch containing the predicate is removed.

Table 2 shows the performance figures of our system on the WSJ and Brown corpora: precision, recall, F_1-measure, and complete proposition accuracy (PP). These figures are compared to the best-performing system in the CoNLL-2005 Shared Task (Punyakanok et al., 2008), referred to as Punyakanok in the table, and the best result currently published (Surdeanu et al., 2007), referred to as Surdeanu. To validate the sanity of the segment creation algorithm, the table also shows the result of applying segment creation to gold-standard syntactic–semantic trees. We see that the two conversion procedures involved (constituent-to-dependency conversion by the CoNLL-2008 Shared Task organizers, and our dependency-to-segment conversion) work satisfactorily although the process is not completely lossless.

Algorithm 2 Segment creation from an argument dependency node.

input Predicate node p, argument node a
if a does not dominate p
  $Y \leftarrow \{n : a \text{ dominates } n\}$
else
  $c \leftarrow$ the child of a that dominates p
  $Y \leftarrow \{n : a \text{ dominates } n\} \setminus \{n : c \text{ dominates } n\}$
end if
$S \leftarrow$ partition of Y into contiguous subsets
return $\{(\text{min-index } s, \text{max-index } s) : s \in S\}$

WSJ	P	R	F1	PP
Our system	82.22	**77.72**	79.90	**57.24**
Punyakanok	82.28	76.78	79.44	53.79
Surdeanu	**87.47**	74.67	**80.56**	51.66
Gold standard	97.38	96.77	97.08	93.20

Brown	P	R	F1	PP
Our system	68.79	61.87	65.15	32.34
Punyakanok	73.38	**62.93**	67.75	32.34
Surdeanu	**81.75**	61.32	**70.08**	**34.33**
Gold standard	97.22	96.55	96.89	92.79

WSJ+Brown	P	R	F1	PP
Our system	80.50	**75.59**	77.97	**53.94**
Punyakanok	81.18	74.92	77.92	50.95
Surdeanu	**86.78**	72.88	**79.22**	49.36
Gold standard	97.36	96.75	97.05	93.15

Table 2: Evaluation with unnormalized segments.

During inspection of the output, we noted that many errors arise from inconsistent punctuation attachment in PropBank/Treebank. We therefore normalized the segments to exclude punctuation at the beginning or end of a segment. The results of this evaluation is shown in Table 3. This table does not include the Surdeanu system since we did not have

access to its output.

WSJ	P	R	F1	PP
Our system	**82.95**	**78.40**	**80.61**	**58.65**
Punyakanok	82.67	77.14	79.81	54.55
Gold standard	97.85	97.24	97.54	94.34

Brown	P	R	F1	PP
Our system	70.84	63.71	67.09	**36.94**
Punyakanok	**74.29**	63.71	**68.60**	34.08
Gold standard	97.46	96.78	97.12	93.41

WSJ+Brown	P	R	F1	PP
Our system	81.39	**76.44**	**78.84**	**55.77**
Punyakanok	**81.63**	75.34	78.36	51.84
Gold standard	97.80	97.18	97.48	94.22

Table 3: Evaluation with normalized segments.

The results on the WSJ test set clearly show that dependency-based SRL systems can rival constituent-based systems in terms of performance – it clearly outperforms the Punyakanok system, and has a higher recall and complete proposition accuracy than the Surdeanu system. We interpret the high recall as a result of the dependency syntactic representation, which makes the parse tree paths simpler and thus the arguments easier to find.

For the Brown test set, on the other hand, the dependency-based system suffers from a low precision compared to the constituent-based systems. Our error analysis indicates that the domain change caused problems with prepositional attachment for the dependency parser – it is well-known that prepositional attachment is a highly lexicalized problem, and thus sensitive to domain changes. We believe that the reason why the constituent-based systems are more robust in this respect is that they utilize a combination strategy, using inputs from two different full constituent parsers, a clause bracketer, and a chunker. However, caution is needed when drawing conclusions from results on the Brown test set, which is only 7,585 words, compared to the 59,100 words in the WSJ test set.

3.2 Dependency-based Evaluation

It has previously been noted (Pradhan et al., 2005) that a segment-based evaluation may be unfavorable to a dependency-based system, and that an evaluation that scores argument *heads* may be more indicative of its true performance. We thus carried out an evaluation using the evaluation script of the CoNLL-2008 Shared Task. In this evaluation method, an argument is counted as correctly identified if its head and label are correct. Note that this is not equivalent to the segment-based metric: In a perfectly identified segment, we may still pick out the wrong head, and if the head is correct, we may infer an incorrect segment. The evaluation script also scores predicate disambiguation performance; we did not include this score since the 2005 systems did not output predicate sense identifiers.

Since CoNLL-2005-style segments have no internal tree structure, it is nontrivial to extract a head. It is conceivable that the output of the parsers used by the Punyakanok system could be used to extract heads, but this is not recommendable because the Punyakanok system is an ensemble system and a segment does not always exactly match a constituent in a parse tree. Furthermore, the CoNLL-2008 constituent-to-dependency conversion method uses a richer structure than just the raw constituents: empty categories, grammatical functions, and named entities. To recreate this additional information, we would have to apply automatic systems and end up with unreliable results.

Instead, we thus chose to find an *upper bound* on the performance of the segment-based system. We applied a simple head-finding procedure (Algorithm 3) to find a set of head nodes for each segment. Since the CoNLL-2005 output does not include dependency information, the algorithm uses gold-standard dependencies and intersects segments with the gold-standard segments. This will give us an upper bound, since if the segment contains the correct head, it will always be counted as correct.

The algorithm looks for dependencies leaving the segment, and if multiple outgoing edges are found, a couple of simple heuristics are applied. We found that the best performance is achieved when selecting only one outgoing edge. "Small clauses," which are split into an object and a predicative complement in the dependency framework, are the only cases where we select two heads.

Table 4 shows the results of the dependency-based evaluation. In the table, the output of the

Algorithm 3 Finding head nodes in a segment.

input Argument segment a
if a overlaps with a segment in the gold standard
  $a \leftarrow$ intersection of a and gold standard
$F \leftarrow \{n$: governor of n outside $a\}$
if $|F| = 1$
  return F
remove punctuation nodes from F
if $|F| = 1$
  return F
if $F = \{n_1, n_2, \ldots\}$ where n_1 is an object and n_2 is the predicative part of a small clause
  return $\{n_1, n_2\}$
if F contains a node n that is a subject or an object
  return $\{n\}$
else
  return $\{n\}$, where n is the leftmost node in F

dependency-based system is compared to the semantic dependency links automatically extracted from the segments of the Punyakanok system.

WSJ	P	R	F1	PP
Our system	**88.46**	**83.55**	**85.93**	**61.97**
Punyakanok	87.25	81.59	84.32	58.17

Brown	P	R	F1	PP
Our system	77.67	**69.63**	73.43	**41.32**
Punyakanok	**80.29**	68.59	**73.98**	37.28

WSJ+Brown	P	R	F1	PP
Our system	**87.07**	**81.68**	**84.29**	**59.22**
Punyakanok	86.94	80.21	83.45	55.39

Table 4: Dependency-based evaluation.

In this evaluation, the dependency-based system has a higher F1-measure than the Punyakanok system on both test sets. This suggests that the main advantage of using a dependency-based semantic role labeler is that it is better at finding the heads of semantic arguments, rather than finding segments. The results are also interesting in comparison to the multi-view system described by Pradhan et al. (2005), which has a reported head F1 measure of 85.2 on the WSJ test set. The figure is not exactly compatible with ours, however, since that system used a different head extraction mechanism.

4 Conclusion

We have described a dependency-based system[1] for semantic role labeling of English in the PropBank framework. Our evaluations show that the performance of our system is close to the state of the art. This holds regardless of whether a segment-based or a dependency-based metric is used. Interestingly, our system has a complete proposition accuracy that surpasses other systems by nearly 3 percentage points. Our system is the first semantic role labeler based only on syntactic dependency that achieves a competitive performance.

Evaluation and comparison is a difficult issue since the natural output of a dependency-based system is a set of semantic links rather than segments, as is normally the case for traditional systems. To handle this situation fairly to both types of systems, we carried out a two-way evaluation: conversion of dependencies to segments for the dependency-based system, and head-finding heuristics for segment-based systems. However, the latter is difficult since no structure is available inside segments, and we had to resort to computing upper-bound results using gold-standard input; despite this, the dependency-based system clearly outperformed the upper bound of the performance of the segment-based system. The comparison can also be slightly misleading since the dependency-based system was optimized for the dependency metric and previous systems for the segment metric.

Our evaluations suggest that the dependency-based SRL system is biased to finding argument heads, rather than argument text snippets, and this is of course perfectly logical. Whether this is an advantage or a drawback will depend on the application – for instance, a template-filling system might need complete segments, while an SRL-based vector space representation for text categorization, or a reasoning application, might prefer using heads only.

In the future, we would like to further investigate whether syntactic and semantic analysis could be integrated more tightly. In this work, we used a sim-

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

plistic loose coupling by means of reranking a small set of complete structures. The same criticisms that are often leveled at reranking-based models clearly apply here too: The set of tentative analyses from the submodules is too small, and the correct analysis is often pruned too early. An example of a method to mitigate this shortcoming is the forest reranking by Huang (2008), in which complex features are evaluated as early as possible.

A Classifier Features

Features Used in Predicate 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 Labeling

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. Form/part-of-speech tag of the left 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↓.

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 3.

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 4.

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

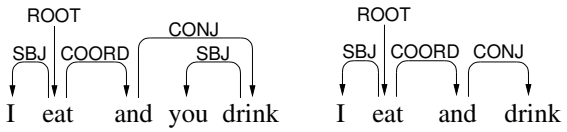

Figure 3: Coordination ambiguity: The subject *I* is in an ambiguous position with respect to *drink*.

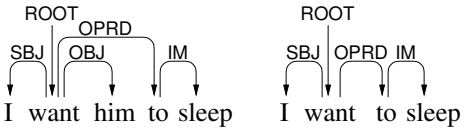

Figure 4: Subject/object control ambiguity: *I* is in an ambiguous position with respect to *sleep*.

References

Collin Baker, Michael Ellsworth, and Katrin Erk. 2007. SemEval task 19: Frame semantic structure extraction. In *Proceedings of SemEval-2007*.

Xavier Carreras and Lluís Màrquez. 2004. Introduction to the CoNLL-2004 shared task: Semantic role labeling. In *Proceedings of CoNLL-2004*.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of CoNLL-2005*.

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

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

Ronan Collobert and Jason Weston. 2007. Fast semantic extraction using a novel neural network architecture. In *Proceedings of ACL-2007*.

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

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

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Daniel Gildea and Martha Palmer. 2002. The necessity of syntactic parsing for predicate argument recognition. In *Proceedings of the ACL-2002*.

Aria Haghighi, Andrew Y. Ng, and Christopher D. Manning. 2005. Robust textual inference via graph matching. In *Proceedings of EMNLP-2005*.

Andrew Hickl, Jeremy Bensley, John Williams, Kirk Roberts, Bryan Rink, and Ying Shi. 2006. Recognizing textual entailment with LCC's GROUNDHOG systems. In *Proceedings of the Second PASCAL Recognizing Textual Entailment Challenge*.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of ACL-2008*.

Richard Johansson and Pierre Nugues. 2008a. Dependency-based syntactic–semantic analysis with PropBank and NomBank. In *Proceedings of the Shared Task Session of CoNLL-2008*.

Richard Johansson and Pierre Nugues. 2008b. The effect of syntactic representation on semantic role labeling. In *Proceedings of COLING-2008*.

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

Ken Litkowski. 2004. Senseval-3 task: Automatic labeling of semantic roles. In *Proceedings of Senseval-3*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

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

Alessandro Moschitti, Paul Morărescu, and Sanda Harabagiu. 2003. Open domain information extraction via automatic semantic labeling. In *Proceedings of FLAIRS*.

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

Jacob Persson, Richard Johansson, and Pierre Nugues. 2008. Text categorization using predicate–argument structures. Submitted.

Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James Martin, and Daniel Jurafsky. 2005. Semantic role labeling using different syntactic views. In *Proceedings of ACL-2005*.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287.

Mihai Surdeanu, Sanda Harabagiu, John Williams, and Paul Aarseth. 2003. Using predicate-argument structures for information extraction. In *Proceedings of ACL-2003*.

Mihai Surdeanu, Lluís Màrquez, Xavier Carreras, and Pere R. Comas. 2007. Combination strategies for semantic role labeling. *Journal of Artificial Intelligence Research*, 29:105–151.

Mihai Surdeanu, 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*.

Reid Swanson and Andrew S. Gordon. 2006. A comparison of alternative parse tree paths for labeling semantic roles. In *Proceedings of COLING/ACL-2006*.

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

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

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of EMNLP-2004*.