Exploration of the LTAG-Spinal Formalism and Treebank for Semantic Role Labeling

Yudong Liu and **Anoop Sarkar**
School of Computing Science
Simon Fraser University
{yudongl,anoop}@cs.sfu.ca

Abstract

LTAG-spinal is a novel variant of traditional Lexicalized Tree Adjoining Grammar (LTAG) introduced by (Shen, 2006). The LTAG-spinal Treebank (Shen et al., 2008) combines elementary trees extracted from the Penn Treebank with Propbank annotation. In this paper, we present a semantic role labeling (SRL) system based on this new resource and provide an experimental comparison with CCGBank and a state-of-the-art SRL system based on Treebank phrase-structure trees. Deep linguistic information such as predicate-argument relationships that are either implicit or absent from the original Penn Treebank are made explicit and accessible in the LTAG-spinal Treebank, which we show to be a useful resource for semantic role labeling.

1 Introduction

Semantic Role Labeling (SRL) aims to identify and label all the arguments for each predicate in a sentence. Specifically, it involves identifying portions of the sentence that represent the predicate's arguments and assigning pre-specified semantic roles to them.

[A0$_{\text{seller}}$ *Ports of Call Inc.*] reached agreements to [V$_{\text{verb}}$ *sell*] [A1$_{\text{thing}}$ *its remaining seven aircraft*] [A2$_{\text{buyer}}$ *to buyers that weren't disclosed*] .

is an example of SRL annotation from the PropBank corpus (Palmer et al., 2005), where the subscripted information maps the semantic roles A0, A1 and A2 to arguments for the predicate *sell* as defined in the PropBank Frame Scheme.

The availability of annotated corpora like PropBank and FrameNet (Fillmore et al., 2001) have provided rapid development of research into SRL (Gildea and Jurafsky, 2002; Gildea and Palmer, 2002; Surdeanu et al., 2003; Chen and Rambow, 2003; Gildea and Hockenmaier, 2003; Xue and Palmer, 2004; Pradhan et al., 2004; Pradhan et al., 2005). The shared tasks in CoNLL-2004 (Carreras and Màrquez, 2004), CoNLL-2005 (Carreras and Màrquez, 2005) and CoNLL-2008 (Surdeanu et al., 2008) were all focused on SRL.

SRL systems (Gildea and Jurafsky, 2002; Gildea and Palmer, 2002) have extensively used features defined over Penn Treebank phrase-structure trees. Other syntactic representations such as CCG derivations (Gildea and Hockenmaier, 2003) and dependency trees (Hacioglu, 2004; Surdeanu et al., 2008) have also been explored. It has been previously noted that LTAG, which has the useful property of *extended domain of locality* (EDL), is well-suited to address the SRL task, c.f. (Chen and Rambow, 2003; Liu and Sarkar, 2007). However, LTAG elementary trees were extracted from the derived parse trees by using Magerman-Collins style head-percolation based heuristic rules (Liu and Sarkar, 2007). The LTAG-spinal Treebank (Shen et al., 2008) provided a corpus of derivation trees where elementary trees were extracted from the Penn Treebank in combination with the Propbank predicate-argument annotation. The LTAG-spinal Treebank can be used to overcome some of the limitations of the previous work on SRL using LTAG: (Liu and Sarkar, 2007) uses LTAG-based features extracted from phrase-structure trees as an additional source of features and combined them with features from a phrase-structure based SRL framework; (Chen and Rambow, 2003) only considers those complement/adjunct semantic roles that can be localized in LTAG elementary trees, which leads to a loss of over 17% instances of semantic roles even from gold-standard trees.

The LTAG-spinal formalism was initially proposed for automatic treebank extraction and statistical parsing (Shen and Joshi, 2005). However, its Propbank-guided treebank extraction process further strengthens the connection between the LTAG-spinal and semantic role labeling. In this paper, we present an SRL system that was built to explore the utility of this new formalism, its Treebank and the output of its statistical parser. Experiments show that our LTAG-spinal based SRL system achieves very high precision on both gold-standard and automatic parses, and significantly outperforms the one using CCGbank. More importantly, it shows that LTAG-spinal is an useful resource for semantic role labeling, with the potential for further improvement.

2 LTAG-spinal, its Treebank and Parsers

This section gives a brief introduction of LTAG-spinal formalism, its Treebank that is extracted with the help of Propbank annotation, and its two statistical parsers that are trained on the Treebank. Predicate-argument relations encoded in the LTAG-spinal treebank will also be discussed to illustrate its compatibility with Propbank and their potential utility for the SRL task.

2.1 LTAG-spinal

The LTAG-spinal formalism (Shen et al., 2008) is a variant of Lexicalized Tree Adjoining Grammar (LTAG) (Abeillé and Rambow, 2001). Compared to traditional LTAG, the two types of elementary trees (e-tree for short), initial and auxiliary trees, are in *spinal* form with no substitution nodes for arguments appearing in the predicate e-tree: a spinal initial tree is composed of a *lexical spine* from the root to the anchor, and nothing else; a spinal auxiliary tree is composed of a *lexical spine* and a *recursive spine* from the root to the foot node. For example, in Figure 1 (from (Shen et al., 2008)), the lexical spine for the auxiliary tree is $B_1, .., B_i, .., B_n$, the recursive spine is $B_1, .., B_i, .., B_1^*$. Two operations *attachment* and *adjunction* are defined in LTAG-spinal where adjunction is the same as adjunction in the traditional LTAG; attachment stems from *sister adjunction* as defined in Tree Insertion Grammar (TIG) (Schabes and Shieber, 1994), which corresponds to the case where the root of an initial tree is taken as a child of another spinal e-tree. The two operations are applied to LTAG-spinal e-tree pairs resulting in an LTAG derivation tree which is similar to a dependency tree (see Figure 2). In Figure 2, e-tree anchored with *continue* is the only auxiliary tree; all other e-trees are initial trees. The arrow is directed from parent to child, with the type of operation labeled on the arc. The operation types are: *att* denotes *attachment* operation; *adj* denotes *adjunction* operation. The sibling nodes may have different landing site along the parent spine. For example, among the child nodes of *stabilize* e-tree, *to* e-tree has VP as landing site; while *even* has S as landing site. Such information, on some level, turns out to be helpful to differentiate the semantic role played by the different child nodes.

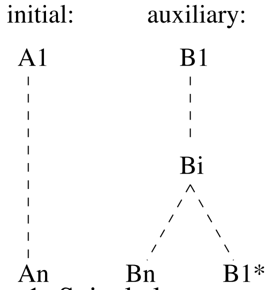

Figure 1: Spinal elementary trees

So far, we can see that in contrast with traditional LTAG where arguments refer to obligatory constituents only, subcategorization frames and argument-adjunct distinction are underspecified in LTAG-spinal. Since argument-adjunct disambiguation is one of the major challenges faced by LTAG treebank construction, LTAG-spinal works around this issue by leaving the disambiguation task for further deep processing, such as semantic role labeling.

LTAG-spinal is weakly equivalent to traditional LTAG with adjunction constraints[1] (Shen, 2006).

The Propbank (Palmer et al., 2005) is an annotated corpus of verb subcategorization and alternations which was created by adding a layer of predicate-argument annotation over the phrase structure trees in the Penn Treebank. The LTAG-spinal Treebank is extracted from the Penn Treebank by exploiting Propbank annotation. Specifically, as described in (Shen et al., 2008), a Penn Treebank syntax tree is taken as an LTAG-spinal derived tree; then information from the Penn Treebank and Propbank is merged using tree transformations. For instance, LTAG predicate coordination and instances of adjunction are recognized using Propbank annotation. LTAG elementary trees are then extracted from the transformed Penn Treebank trees recursively, using the Propbank annotation and a Magerman-Collins style head percolation table.

This guided extraction process allows syntax and semantic role information to be combined in LTAG-spinal derivation trees. For example, the

[1] null adjunction (NA), obligatory adjunction (OA) and selective adjunction (SA)

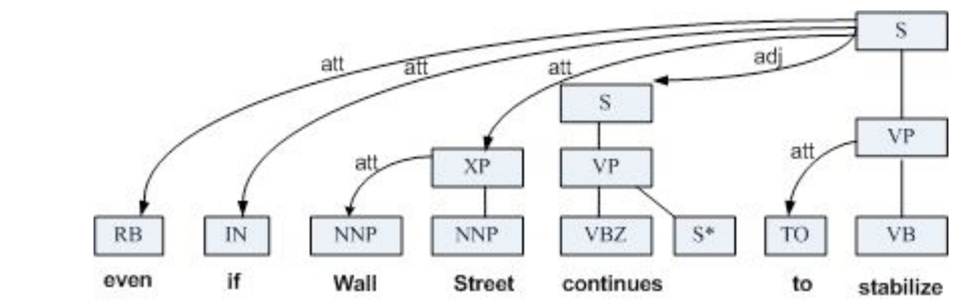

Figure 2: An example of LTAG-spinal sub-derivation tree, from LTAG-spinal Treebank Section 22

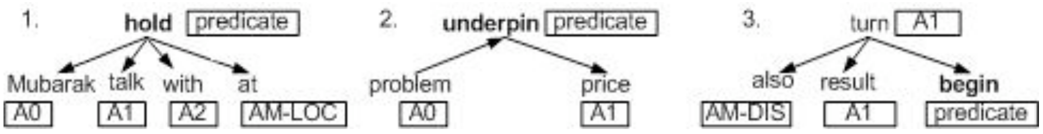

Figure 3: Three examples of LTAG-spinal derivation trees where predicates and their Propbank style argument labels are given. These examples are from LTAG-spinal Treebank Section 22.

Penn Treebank does not differentiate raising verbs and control verbs, however, based on the Propbank information, LTAG-spinal makes this distinction explicit. Thus, the error of taking a subject argument which is not semantically an argument of the raising verb can be avoided. Another property of LTAG-spinal Treebank extraction lies in the flexibility and simplicity of the treatment of predicate coordination (see (Shen et al., 2008)). Figure 3 shows three examples of Propbank annotation as decorations over the LTAG-spinal derivation trees. In each derivation tree, each node is associated with LTAG-spinal e-trees. Each argument (A0, A1, etc.) is referred to as A and the predicate is called P. In most cases, the argument is found locally in the derivation tree due to the extended domain of locality in e-trees. Thus, most arguments are identified by the pattern $P \rightarrow A$ or $P \leftarrow A$. The next section contains a discussion of such patterns in more detail.

Two statistical parsers have been developed by Libin Shen specifically for training on the LTAG-spinal treebank: a left-to-right incremental parser (Shen and Joshi, 2005) and a bidirectional incremental parser (Shen and Joshi, 2008). If one compares the output of these two parsers, the left-to-right parser produces full LTAG-spinal derivation trees (including all the information about specific elementary trees used in the derivation and the attachment information within the e-trees) while the bidirectional parser produces derivation trees without information about elementary trees or attachment points (similar to output from a dependency parser). In this paper, we use the left-to-right incremental parser for its richer output because our SRL system uses feature functions that use information about the elementary trees in the derivation tree and the attachment points between e-trees. The landing site of child node along the parent spine is useful for identifying different types of arguments in SRL. For example, assume the parent spine is “S-VP-VB-anchor” (the root label is S, and “anchor” is where the lexical item is inserted). Along with direction information, the landing site label “S” is likely to be a good indicator for argument A0 (subject) while the landing site label “VP” could be a good indicator for “A1” (object). In this sense, the incremental left-to-right parser is preferable for semantic role labeling. However, having been developed earlier than the bidirectional parser, the incremental parser obtains 1.2% less in dependency accuracy compared to the bidirectional parser (Shen and Joshi, 2008).

2.2 Predicate-argument relations in the LTAG-spinal Treebank

The Propbank-guided extraction process for LTAG-spinal treebank naturally creates a close connection between these two resources. To examine the compatibility of the LTAG-spinal Treebank with Propbank, (Shen et al., 2008) provides the frequency for specific types of paths from the predicate to the argument in the LTAG-spinal derivation trees from the LTAG-spinal Treebank. The 8 most frequent patterns account for 95.5% of the total predicate-argument pairs of the LTAG-spinal Treebank, of which 88.4% are directly connected pairs. These statistics not only provide em-

	Path Pattern	Number	Percent
1	P→A	8294	81.3
2	P←A, V←A	720	7.1
3	P←Px→A	437	4.3
4	P←Coord→Px→A	216	2.1
5	P←Ax←Py→A	84	0.82
6	P←Coord←Px→A	40	0.39
7	P←Px←Py→A	13	0.13
total recovered w/ patterns		9804	96.1
total		10206	100.0

Table 1: Distribution of the 7 most frequent predicate-argument pair patterns in LTAG-spinal Treebank Section 22. P: predicate, A: argument, V: modifying verb, $Coord$: predicate coordination.

pirical justification for the notion of the extended domain of locality (EDL) in LTAG-spinal (Shen et al., 2008), they also provide motivation to explore this Treebank for the SRL task.

We collected similar statistics from Treebank Section 22 for the SRL task, shown in Table 1, where 7 instead of 8 patterns suffice in our setting. Each pattern describes one type of P(redicate)-A(rgument) pair with respect to their dependency relation and distance in the LTAG-spinal derivation tree. The reason that we combine the two patterns P←A and V←A into one is that from SRL perspective, they are equivalent in terms of the dependency relation and distance between the predicate. Each token present in the patterns, such as P, Px, Py, V, A, Ax and Coord, denotes a spinal e-tree in the LTAG-spinal derivation tree.

To explain the patterns more specifically, take the LTAG-spinal sub-derivation tree in Figure 2 as an example, Assume P(redicate) in question is *stabilize* then (*stabilize* → *even*), (*stabilize* → *if*), (*stabilize* → *Street*), (*stabilize* → *continue*), (*stabilize* → *to*) all belong to pattern 1; but only (*stabilize* → *Street*) is actual predicate-argument pair. Similarly, when take *continue* as P, the predicate-argument pair (*continue* ← *stabilize*) belongs to pattern 2, where *stabilize* corresponds to A(rgument) in the pattern; (*continue*, *Street*) in (*Street* ← *stabilize* → *continue*) is an example of pattern 3, where *stabilize* corresponds to Px and *Street* corresponds to A in the pattern 3 schema. Pattern 4 denotes the case where argument (A) is shared between coordinated predicates (P and Px); The main difference of pattern 5-7 exists where the sibling node of A(rgument) is categorized into: predicate (Px) in pattern 7, predicate coordination node (Coord) in pattern 6 and others (Ax) in pattern 5. We will retain this difference instead of merging it since the semantic relation between P and A varies based on these differences. Example sentences for other (rarer) patterns can be found in (Shen et al., 2008).

3 LTAG-spinal based SRL System Description

In this section, we describe our LTAG-spinal based SRL system. So far, we have studied LTAG-spinal formalism, its treebank and parsers. In particular, the frequency distribution of the seven most seen predicate-argument pair patterns in LTAG-spinal Treebank tells us that predicate-argument relationships typical to semantic role labeling are often local in LTAG-spinal derivation trees.

Pruning, argument identification and argument classification – the 3-stage architecture now standard in SRL systems is also used in this paper. Specifically, for the sake of efficiency, nodes with high probability of being NULL (non-argument) should be filtered at the beginning; usually filtering is done based on some heuristic rules; after the pruning stage, argument identification takes place with the goal of classifying the pruning-survival nodes into argument and non-argument; for those nodes that have been classified as arguments, argument classification component will further label them with different argument types, such as A0, A1, etc. Argument identification and classification are highly ambiguous tasks and are usually accomplished using a machine learning method.

For our LTAG-spinal based SRL system, we first collect the argument candidates for each predicate from the LTAG-spinal derivation tree. For each candidate, features are extracted to capture the predicate-argument relations. Binary classifiers for identification and classification are trained using SVMs and combined in a one-vs-all model. The results are evaluated using precision/recall/f-score.

3.1 Candidate Locations for Arguments

In SRL systems that perform role labeling of constituents in a phrase-structure tree, statistics show that after pruning, ∼98% of the SRL argument nodes are retained in the gold-standard trees in the Penn Treebank, which provides a high upper-bound for the recall of the SRL system. Pruning away unnecessary nodes using a heuristic makes learning easier as well, as many of the false positives are pruned away leading to a more balanced binary classification problem during the semantic role identification and classification steps. We need a similar heuristic over LTAG-spinal nodes that will have high coverage with respect to SRL arguments and provide a high upper-bound for recall.

As previously shown that the seven most frequent predicate-argument pair patterns that are used to describe the specific types of paths from the predicate to the argument account for ∼96% of the total number of predicate-argument pairs in the LTAG-spinal Treebank. These patterns provide a natural candidate selection strategy for our SRL.

Table 2 shows a similar oracle test applied to the output of the LTAG-spinal parser on Section 22. The total drop in oracle predicate-argument identifiation drops 10.5% compared to gold-standard trees. 9.8% is lost from patterns 1 and 2. If exclude those pairs that belong to pattern i in treebank but belong to pattern j $(i \neq j)$ in automatic parses (so the pattern exists but is the wrong one for that constituent), the number drops to 81.6% from 85.6%. This indicates that in terms of the impact of the syntactic parser errors for SRL, the LTAG-spinal parser will suffer even more than the phase structure parser. An alternative is to exhaustively search for predicate-argument pairs without considering patterns, which we found introduces too much noise in the learner to be feasible. Thus, the predicate-argument pairs selected through this phase are considered as argument candidates for our SRL system.

3.2 Features

Based on the patterns, features are defined on predicate-argument pairs from LTAG derivation tree, mainly including *predicate e-trees*, *argument e-trees*, *intermediate e-trees* and their "topological relationships" such as *operation, spine node, relative position* and *distance*. The following are the specific features used in our classifiers:

Features from predicate e-tree and its variants predicate lemma, POS tag of predicate, predicate voice, spine of the predicate e-tree, 2 variants of predicate e-tree: replacing anchor in the spine with predicate lemma, replacing anchor POS in the spine with voice. In Figure 2, if take *stabilize* as predicate, these two variants are *S-VP-VB-stabilize* and *S-VP-VB-active* respectively.

Features from argument e-tree and its variants argument lemma, POS tag of argument, Named Entity (NE) label of the argument, spine of the argument e-tree, 2 variants of argument e-tree: replacing anchor in the spine with argument lemma, replacing anchor POS with NE label if any, label of root node of the argument spine. In Figure 2, if take *stabilize* as predicate, and *Street* as argument, the two variants are *XP-NNP-street* and *XP-ORGANIZATION*[2] respectively.

PP content word of argument e-tree if the root label of the argument e-tree is PP, anchor of the last daughter node. NE variant of this feature: replace its POS with the NE label if any.

Features from the spine node (SP1) spine node is the landing site between predicate e-tree and argument e-tree. Features include the index along the host spine[3], label of the node, operation involved (*att* or *adj*).

Relative position of predicate and argument in the sentence: before/after.

Order of current child node among its siblings. In pattern 1, predicate e-tree is parent, and argument e-tree is child. This feature refers to the order of argument e-tree among its siblings nodes (with predicate e-tree as parent).

Distance of predicate e-tree and argument tree in the LTAG derivation tree: For example, for pattern 1 and 2, the distance has value 0; for pattern 3, the distance has value 1.

Pattern ID valued 1-7. (see Table 1 and Table 2)

Combination of position and pattern ID, combination of distance and pattern ID, combination of

	Path Pattern	Number	Percent
1	P→A	7441	72.9
2	P←A, V←A	583	5.7
3	P←Px→A	384	3.8
4	P←Coord→Px→A	180	1.76
5	P←Ax←Py→A	75	0.73
6	P←Coord←Px→A	48	0.47
7	P←Px←Py→A	22	0.21
total recovered w/ patterns		8733	85.6
total		10206	100.0

Table 2: Distribution of the 7 patterns in LTAG-spinal parser output for Section 22.

[2]XP-NNP is a normalized e-tree form used in (Shen et al., 2008) for efficiency and to avoid the problem of sparse data over too many e-trees.

[3]it can either be predicate e-tree or argument e-tree. For example, for pattern P←A, the A(rgument) e-tree is the host spine.

position and order.

Features from intermediate predicate e-tree same features as predicate e-tree features.

Features from spine node of intermediate predicate e-tree and argument e-tree (SP2) for predicate-argument pairs of pattern 3-7. These features are similar to the SP1 features but instead between intermediate predicate e-tree and argument e-tree.

Relative position between predicate e-tree and intermediate e-tree.

Combination relative positions of argument e-tree and intermediate predicate e-tree + relative position of argument e-tree and predicate e-tree.

The features listed above are used to represent each candidate constituent (or node) in the LTAG-spinal derivation tree in training and test data. In both cases, we identify SRLs for nodes for each predicate. In training each node comes with the appropriate semantic role label, or NULL if it does not have any (for the predicate). In test data, we first identify nodes as arguments using these features (ARG v.s. NULL classification) and then classify a node identified as an argument with the particular SRL using one-vs-all binary classification.

4 Experiments

4.1 Data Set

Following the usual convention for parsing and SRL experiments, LTAG-spinal Treebank Section 2-21 is used for training and Section 23 for testing. Propbank argument set is used which includes numbered arguments A0 to A5 and 13 adjunct-like arguments. 454 sentences in the Penn Treebank are skipped from the LTAG-spinal Treebank (Shen et al., 2008)[4], which results in 115 predicate-argument pairs ignored in the test set.

We applied SVM-light (Joachims, 1999) with default linear kernel to feature vectors. 30% of the training samples are used to fine tune the regularization parameter c and the loss-function cost parameter j for both argument identification and classification. With parameter validation experiments, we set $c = 0.1$ and $j = 1$ for {A0, AM-NEG}, $c = 0.1$, $j = 2$ for {A1, A2, A4, AM-EXT} and $c = 0.1$ and $j = 4$ for the rest.

For comparison, we also built up a standard 3-stage phrase-structure based SRL system, where exactly the same data set[5] is used from 2004 February release of the Propbank. SVM-light with linear kernel is used to train on a standard feature set (Xue and Palmer, 2004). The Charniak and Johnson parser (2006) is used to produce the automatic parses. Note that this phrase-structure based SRL system is state-of-the-art and we have included all the features proposed in the literature that use phrase-structure trees. This system obtains a higher SRL accuracy which can be improved only by using global inference and other ways (such as using multiple parsers) to improve the accuracy on automatic parses.

4.2 Results

We compared our LTAG-spinal based SRL system with phrase-structure based one (see the description in earlier sections), for argument identification and classification. In order to analyze the impact of errors in syntactic parsers, results are presented on both gold-standard trees and automatic parses. Based on the fact that nearly 97% e-trees that correspond to the core arguments[6] belong to pattern 1 and 2, which accounts for the largest portion of argument loss in automatic parses, the classification results are also given for these core arguments. We also compare with the CCG-based SRL presented in (Gildea and Hockenmaier, 2003)[7], which has a similar motivation as this paper, except they use the Combinatory Categorial Grammar formalism and the CCGBank syntactic Treebank which was converted from the Penn Treebank.

Scoring strategy To have a fair evaluation of arguments between the LTAG-spinal dependency parse and the Penn Treebank phrase structure, we report the *root/head-word* based scoring strategy for performance comparison, where a case is counted as positive as long as the root of the argument e-tree is correctly identified in LTAG-spinal and the head word of the argument constituent is correctly identified in phrase structure. In contrast, boundary-

[4]Based on (Shen et al., 2008), the skipped 454 sentences amount to less than 1% of the total sentences. 314 of these 454 sentences have gapping structures. Since PTB does not annotate the trace of deleted predicates, additional manual annotation is required to handle these sentences. For the rest of the 146 sentences, abnormal structures are generated due to tagging errors.

[5]The same 454 sentences are ignored.

[6]A0, A1, A2, A3, A4, A5

[7]Their data includes the 454 sentences. However, the missing 115 predicate-argument pairs account for less than 1% of the total number of predicate-argument pairs in the test data, so even if we award these cases to the CCGBank system the system performance gap still remains.

based scoring is more strict in that the string span of the argument must be correctly identified in identification and classification.

Results from using gold standard trees Table 3 shows the results when gold standard trees are used. We can see that with gold-standard derivations, LTAG-spinal obtains the highest precision on identification and classification; it also achieves a competitive f-score (highest f-score for identification) with the recall upper-bound lower by 2-3% than phrase-structure based SRL. However, the recall gap between the two SRL systems gets larger for classification compared to identification[8], which is due to the low recall that is observed with our LTAG-spinal based SRL based on our current set of features. If compare the difference between the root/head-word based score and the boundary based score in the 3 scenarios, we notice that the difference reflects the discrepancy between the argument boundaries. It is not surprising to see that phrase-structure based one has the best match. However, CCGBank appears to have a large degree of mismatch. In this sense, root/head word based scoring provides fair comparison between LTAG-spinal SRL system and the CCGBank SRL system.

Recent work (Boxwell and White, 2008) changes some structures in the CCGBank to correspond more closely with the Probbank annotations. They also resolve split arguments that occur in Propbank and add these annotations into a revised version of the CCGBank. As a result they show that the *oracle* f-score improves by over 2 points over the (Gildea and Hockenmaier, 2003) oracle results for the numbered arguments only (A0, . . ., A5). It remains an open question whether a full SRL system based on a CCG parser trained on this new version of the CCGBank will be competitive against the LTAG-spinal based and phrase-structure based SRL systems.

Results from using automatic parses Table 4 shows the results when automatic parses are used. With automatic parses, the advantage of LTAG-spinal in the precision scores still exists: giving a higher score in both identification and core argument classification; only 0.5% lower for full argument classification. However, with over 6% difference in upper-bound of recall ($\leq$85.6% from LTAG-spinal; $\sim$91.7% from Charniak's parser), the gap in recall becomes larger: increased to $\sim$10% in automatic parses from $\sim$6% in gold-standard trees.

The identification result is not available for CCG-based SRL. In terms of argument classification, it is significantly outperformed by the LTAG-spinal based SRL. In particular, it can be seen that the LTAG-spinal parser performs much better on argument boundaries than CCG-based one.

One thing worth mentioning is that since neither the LTAG-spinal parser nor Charniak's parser provides trace (empty category) information in their output, no trace information is used for LTAG-spinal based SRL or the phrase-structure based SRL even though it is available in their gold-standard trees.

5 Conclusion and Future Work

With a small feature set, the LTAG-spinal based SRL system described in this paper provides the highest precision in almost all the scenarios, which indicates that the shallow semantic relations, e.g., the predicate-argument relations that are encoded in the LTAG-spinal Treebank are useful for SRL, especially when compared to the phrase structure Penn Treebank. (Shen et al., 2008) achieves an f-score of 91.6% for non-trace SRL identification on the entire Treebank by employing a simple rule-based system, which also suggested this conclusion. In other words, there is a tighter connection between the syntax and semantic role labels in the LTAG-spinal representation.

However, in contrast to the high precision, the recall performance of LTAG-spinal based SRL needs a further improvement, especially for the argument classification task. From SRL perspective, on one hand, this may be due to the pattern-based candidate selection, which upper-bounds the number of predicate-argument pairs that can be recovered for SRL; on the other hand, it suggests that the features for argument classification need to be looked at more carefully, compared to the feature selection for argument identification, especially for A2 and A3 (as indicated by our error analysis on the results on the development set). A possible solution is to customize a different feature set for each argument type during classification, especially for contextual information.

Experiments show that when following the pipelined architecture, the performance of LTAG-based SRL is more severely degraded by the syntactic parser, compared to the SRL using phrase

[8]no NULL examples are involved when training for argument classification.

Identification	gold-standard trees (p/r/f%)		
Scoring	LTAG	phrase	CCG
Root/head-word	**96.0**/92.1/**94.0**	93.0/94.0/93.5	n/a
classification (core)	gold-standard trees (p/r/f%)		
Scoring	LTAG	phrase	CCG
Root/head-word	**90.6**/83.4/86.9	87.2/88.4/87.8	82.4/78.6/80.4
classification (full)	gold-standard trees (p/r/f%)		
Scoring	LTAG	phrase	CCG
Root/head-word	**88.2**/81.7/84.8	86.1/87.1/86.6	76.3/67.8/71.8
Boundary	**87.4**/81.0/84.1	86.0/87.0/86.5	67.5/60.0/63.5

Table 3: Using gold standard trees: comparison of the three SRL systems for argument identification, core and full argument classification

Identification	automatic parses (p/r/f%)		
Scoring	LTAG	phrase	CCG
Root/head-word	**85.8**/80.0/82.8	85.8/87.7/86.7	n/a
classification (core)	automatic parses (p/r/f%)		
Scoring	LTAG	phrase	CCG
Root/head-word	**81.0**/71.5/76.0	80.1/82.8/81.4	76.1/73.5/74.8
classification (full)	automatic parses (p/r/f%)		
Scoring	LTAG	phrase	CCG
Root/head-word	78.0/70.0/73.7	78.5/80.3/79.4	71.0/63.1/66.8
Boundary	72.3/65.0/68.5	73.8/75.5/74.7	55.7/49.5/52.4

Table 4: Using automatic parses: comparison of the three SRL systems for argument identification, core and full argument classification

structure and CCG formalism. Even though the left-to-right statistical parser that was trained and evaluated on the LTAG-spinal Treebank achieves an f-score of 89.3% for dependencies on Section 23 of this treebank (Shen and Joshi, 2005), the SRL that used this output is worse than expected. An oracle test shows that via the same 7 patterns, only 81.6% predicate-argument pairs can be recovered from the automatic parses, which is a big drop from 96.1% when we use the LTAG-spinal Treebank trees. Parser accuracy is high overall, but needs to be more accurate in recovering the dependencies between predicate and argument.

Based on the observation that the low recall occurs not only to the SRL when the automatic parses are used but also when the gold trees are used, we would expect that a thorough error analysis and feature calibrating can give us a better idea in terms of how to increase the recall in both cases.

In on-going work, we also plan to improve the dependency accuracy for predicate and argument dependencies by using the SRL predictions as feedback for the syntactic parser. Our hypothesis is that this approach combined with features that would improve the recall numbers would lead to a highly accurate SRL system.

As a final note, we believe that our effort on using LTAG-spinal for SRL is a valuable exploration of the LTAG-spinal formalism and its Treebank resource. We hope our work will provide useful information on how to better utilize this formalism and the Treebank resource for semantic role labeling.

Acknowledgements

We would like to thank Aravind Joshi and Lucas Champollion for their useful comments and for providing us access to the LTAG-spinal Treebank. We would especially like to thank Libin Shen for providing us with the LTAG-spinal statistical parser for our experiments and for many helpful comments.

References

A. Abeillé and O. Rambow, editors. 2001. *Tree Adjoining Grammars: Formalisms, Linguistic Analysis and Processing*. Center for the Study of Language and Information.

Stephen A. Boxwell and Michael White. 2008. Projecting propbank roles onto the ccgbank. In *LREC-2008*.

X. Carreras and L. Màrquez. 2004. Introduction to the CoNLL-2004 Shared Task. In *CoNLL-2004*.

X. Carreras and L. Màrquez. 2005. Introduction to the CoNLL-2005 Shared Task. In *CoNLL-2005*.

J. Chen and O. Rambow. 2003. Use of deep linguistic features for the recognition and labeling of semantic arguments. In *EMNLP-2003*.

C.J. Fillmore, C. Wooters, and C.F. Baker. 2001. Building a large lexical databank which provides deep semantics. In *PACLIC15-2001*.

D. Gildea and J. Hockenmaier. 2003. Identifying semantic roles using combinatory categorial grammar. In *EMNLP-2003*.

D. Gildea and D. Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 58(3):245–288.

D. Gildea and M. Palmer. 2002. The necessity of parsing for predicate argument recognition. In *ACL-2002*.

K. Hacioglu. 2004. Semantic role labeling using dependency trees. In *COLING-2004*.

T. Joachims. 1999. Making large-scale svm learning practical. *Advances in Kernel Methods - Support Vector Machines*.

Y. Liu and A. Sarkar. 2007. Experimental evaluation of LTAG-based features for semantic role labeling. In *EMNLP-2007*.

M. Palmer, D. Gildea, and P. Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1).

S. Pradhan, W. Ward, K. Hacioglu, , J. H. Martin, and D. Jurafsky. 2004. Shallow Semantic Parsing Using Support Vector Machines. In *HLT-NAACL-2004*.

S. Pradhan, W. Ward, K. Hacioglu, , J. H. Martin, and D. Jurafsky. 2005. Semantic role labeling using different syntactic views. In *ACL-2005*.

Yves Schabes and Stuart M. Shieber. 1994. An alternative conception of tree-adjoining derivation. *Computational Linguistics*, 20(1):91–124.

L. Shen and Aravind Joshi. 2005. Incremental ltag parsing. In *HLT-EMNLP-2005*.

L. Shen and A. Joshi. 2008. Ltag dependency parsing with bidirectional incremental construction. In *EMNLP-2008*.

L. Shen, L. Champollion, and A. Joshi. 2008. Ltag-spinal and the treebank: A new resource for incremental, dependency and semantic parsing. *Language Resources and Evaluation*, 42(1):1–19.

L. Shen. 2006. *Statistical LTAG Parsing*. Ph.D. thesis, University of Pennsylvania.

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

M. Surdeanu, R. Johansson, A. Meyers, L. Màrquez, and J. Nivre. 2008. The conll 2008 shared task on joint parsing of syntactic and semantic dependencies. In *CoNLL-2008*.

N. Xue and M. Palmer. 2004. Calibrating features for semantic role labeling. In *EMNLP-2004*.