Experimental Evaluation of LTAG-based Features for Semantic Role Labeling

Yudong Liu and **Anoop Sarkar**
Simon Fraser University
Burnaby, BC, Canada
{yudongl,anoop}@cs.sfu.ca

Abstract

This paper proposes the use of Lexicalized Tree-Adjoining Grammar (LTAG) formalism as an important additional source of features for the Semantic Role Labeling (SRL) task. Using a set of one-vs-all Support Vector Machines (SVMs), we evaluate these LTAG-based features. Our experiments show that LTAG-based features can improve SRL accuracy significantly. When compared with the best known set of features that are used in state of the art SRL systems we obtain an improvement in F-score from 82.34% to 85.25%.

1 Introduction

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

[$\mathrm{A0}_{\textit{seller}}$ *Ports of Call Inc.*] reached agreements to [$\mathrm{V}_{\mathrm{verb}}$ *sell*] [$\mathrm{A1}_{\mathrm{thing}}$ *its remaining seven aircraft*] [$\mathrm{A2}_{\mathrm{buyer}}$ *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, A2 to arguments for the predicate *sell* as defined in the PropBank Frame Scheme. For SRL, high accuracy has been achieved by:
(i) proposing new types of features (see Table 1 in Section 3 for previously proposed features),
(ii) modeling the predicate frameset by capturing dependencies between arguments (Gildea and Jurafsky, 2002; Pradhan et al., 2004; Toutanova et al., 2005; Punyakanok et al., 2005a),
(iii) dealing with incorrect parser output by using more than one parser (Pradhan et al., 2005b).

Our work in this paper falls into category (i). We propose several novel features based on Lexicalized Tree Adjoining Grammar (LTAG) derivation trees in order to improve SRL performance. To show the usefulness of these features, we provide an experimental study comparing LTAG-based features with the standard set of features and kernel methods used in state-of-the-art SRL systems. The LTAG formalism provides an extended domain of locality in which to specify predicate-argument relationships and also provides the notion of a derivation tree. These two properties of LTAG make it well suited to address the SRL task.

SRL feature extraction has relied on various syntactic representations of input sentences, such as syntactic chunks (Hacioglu et al., 2004) and full syntactic parses (Gildea and Jurafsky, 2002). In contrast with features from shallow parsing, previous work (Gildea and Palmer, 2002; Punyakanok et al., 2005b) has shown the necessity of full syntactic parsing for SRL. In order to generalize the *path* feature (see Table 1 in Section 3) which is probably the most salient (while being the most data sparse) feature for SRL, previous work has extracted features from other syntactic representations, such as CCG derivations (Gildea and Hockenmaier, 2003) and dependency trees (Hacioglu, 2004) or integrated features from different parsers (Pradhan et al., 2005b). To avoid explicit feature engineering on trees, (Moschitti, 2004) used convolution kernels on selective portions of syntactic trees. In this paper, we also compare our work with tree kernel based methods.

Most SRL systems exploit syntactic trees as the main source of features. We would like to take this one step further and show that using LTAG deriva-

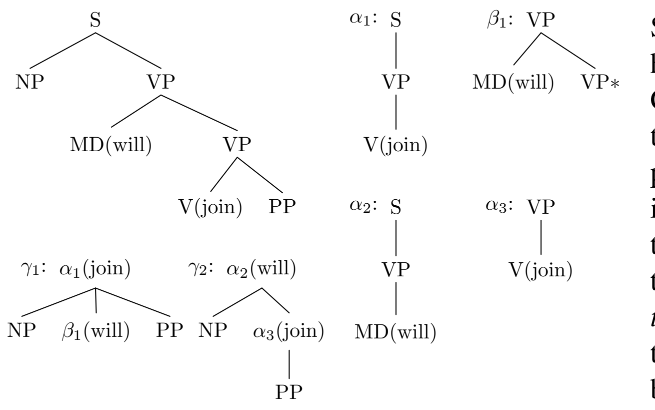

Figure 1: A parse tree schematic, and two plausible LTAG derivation trees for it: derivation tree γ_1 uses elementary trees α_1 and β_1 while γ_2 uses α_2 and α_3.

tion trees as an additional source of features can improve both argument identification and classification accuracy in SRL.

2 Using LTAG-based Features in SRL

We assume some familiarity with Lexicalized Tree-Adjoining Grammar (LTAG); (Joshi and Schabes, 1997) is a good introduction to this formalism. A LTAG is defined to be a set of lexicalized *elementary trees* (etree for short), of which there are two types, *initial trees* and *auxiliary trees*. Typically etrees can be composed through two operations into parse trees, *substitution* and *adjunction*. We use *sister adjunction* which is commonly used in LTAG statistical parsers to deal with the relatively flat Penn Treebank trees (Chiang, 2000). The tree produced by composing the etrees is the *derived/parse tree* and the tree that records the history of composition is the *derivation tree*.

A reasonable way to define SRL features is to provide a strictly local dependency (i.e. within a single etree) between predicate and argument. There have been many different proposals on how to maintain syntactic locality (Xia, 1999; Chen and Vijay-Shanker, 2000) and SRL locality (Chen and Rambow, 2003; Shen and Joshi, 2005) when extracting LTAG etrees from a Treebank. These proposed methods are exemplified by the derivation tree γ_1 in Fig. 1. However, in most cases they can only provide a local dependency between predicate and argument for 87% of the argument constituents (Chen and Rambow, 2003), which is too low to provide high SRL accuracy. In LTAG-based statistical parsers, high accuracy is obtained by using the Magerman-Collins head-percolation rules in order to provide the etrees (Chiang, 2000). This method is exemplified by the derivation tree γ_2 in Fig. 1. Comparing γ_1 with γ_2 in Fig. 1 and assuming that *join* is the predicate and the *NP* is the potential argument, the path feature as defined over the LTAG *derivation tree* γ_2 is more useful for the SRL task as it distinguishes between main clause and non-finite embedded clause predicates. This alternative derivation tree also exploits the so-called *extended domain of locality* (Joshi and Schabes, 1997) (the examples in Section 2.1 show this clearly). In this paper, we crucially rely on features defined on LTAG derivation trees of the latter kind. We use polynomial kernels to create combinations of features defined on LTAG derivation trees.

2.1 LTAG-based Feature Extraction

In order to create training data for the LTAG-based features, we convert the Penn Treebank phrase structure trees into LTAG derivations. First, we prune the Treebank parse tree using certain constraints. Then we decompose the pruned parse trees into a set of LTAG elementary trees and obtain a derivation tree. For each constituent in question, we extract features from the LTAG derivation tree. We combine these features with the standard features used for SRL and train an SVM classifier on the combined LTAG derivation plus SRL annotations from the PropBank corpus.

For the test data, we report on results using the gold-standard Treebank data, and in addition we also report results on automatically parsed data using the Charniak parser (Charniak, 2000) as provided by the CoNLL 2005 shared task. We did this for three reasons: (i) our results are directly comparable to those who have used the Charniak parses distributed with the CoNLL 2005 data-set; (ii) we avoid the possibility of a better parser identifying a larger number of argument constituents and thus leading to better results, which is orthogonal to the discriminative power of our proposed LTAG-based features; and (iii) the quality of LTAG derivation trees depends indirectly on the quality of head dependencies recovered by the parser and it is a well-known folklore result (see Table 3 in (McDonald et al., 2005)) that applying the head-percolation heuristics on parser output produces better dependencies when compared to dependencies directly recovered by the parser (whether the parser is an LTAG parser or a lexicalized PCFG parser).

2.1.1 Pruning Parse Trees

Given a parse tree, the pruning component identifies the predicate in the tree and then only admits those nodes that are sisters to the path from the predicate to the root. It is commonly used in the SRL community (cf. (Xue and Palmer, 2004)) and our experiments show that 91% of the SRL targets can be recovered despite this aggressive pruning. We make two enhancements to the pruned Propbank tree: we enrich the sister nodes with head information, a part-of-speech tag and word pair: $\langle t, w \rangle$ and PP nodes are expanded to include the NP complement of the PP (including head information). The target SRL node is still the PP. Figure 2 is a pruned parse tree for a sentence from the PropBank.

2.1.2 Decompositions of Parse Trees

After pruning, the pruned tree is decomposed around the predicate using standard head-percolation based heuristic rules[1] to convert a Treebank tree into an LTAG derivation tree. Figure 3 shows the resulting etrees after decomposition. Figure 4 is the derivation tree for the entire pruned tree. Each node in this derivation tree represents an etree in Figure 3. In our model we make an independence assumption that each SRL is assigned to each constituent independently, conditional only on the path from the *predicate etree* to the *argument etree* in the derivation tree. Different etree siblings in the LTAG derivation tree do not influence each other in our current models.

2.1.3 LTAG-based Features

We defined 5 LTAG feature categories: *predicate etree*-related features (**P** for short), *argument etree*-related features (**A**), *subcategorization*-related features (**S**), *topological relation*-related features (**R**), *intermediate etree*-related features (**I**). Since we consider up to 6 intermediate etrees between the predicate and the argument etree, we use *I*-1 to *I*-6 to represent these 6 intermediate trees respectively.

[1]using http://www.isi.edu/∼chiang/software/treep/treep.html

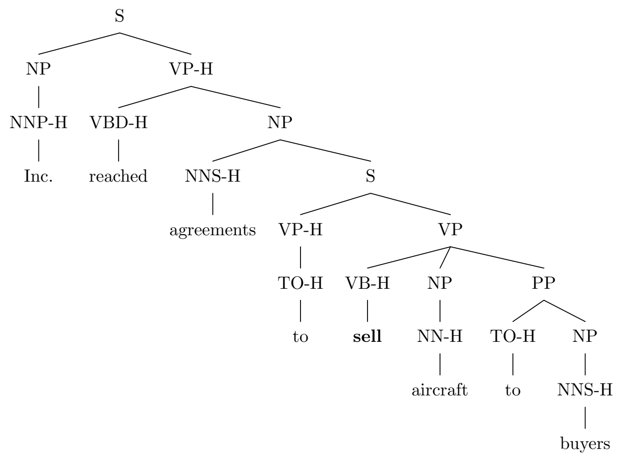

Figure 2: The pruned tree for the sentence "*Ports of Call Inc. reached agreements to **sell** its remaining seven aircraft to buyers that weren't disclosed.*"

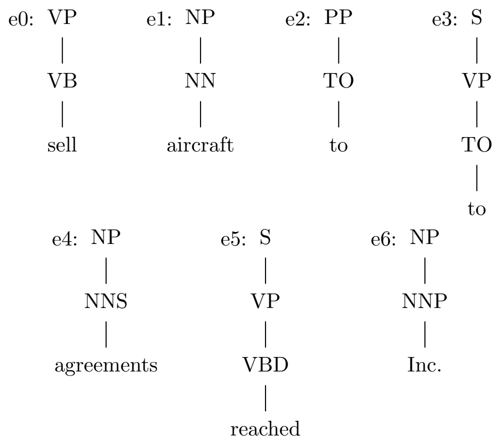

Figure 3: Elementary trees after decomposition of the pruned tree.

Category P: Predicate etree & its variants *Predicate etree* is an etree with predicate, such as $e0$ in Figure 3. This new feature complements the predicate feature in the standard SRL feature set. One variant is to remove the predicate lemma. Another variant is a combination of *predicate tree w/o predicate lemma&POS* and *voice*. In addition, this variant combined with predicate lemma comprises another new feature. In the example, these three variants are *(VP(VB))* and *(VP)_active* and *(VP)_active_sell* respectively.

Category A: Argument etree & its variants Analogous to the predicate etree, the *argument etree* is an etree with the target constituent and its head. Similar

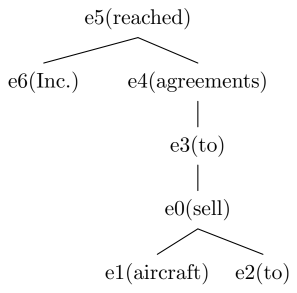

Figure 4: LTAG derivation tree for Figure 2.

to predicate etree related features, argument etree, argument etree with removal of head word, combination of *argument etree w/o head POS&head word* and *head Named Entity (NE) label (if any)* are considered. For example, in Figure 3, these 3 features for $e6$ are $e6$, *(NP(NNP))* and *(NP)_LOC* with head word "Inc." having NE label "LOC".

Category S: Index of current argument etree in subcat frame of predicate etree Sub-categorization is a standard feature that denotes the immediate expansion of the predicate's parent. For example, it is *V_NP_PP* for predicate *sell* in the given sentence. For argument etree $e1$ in Figure 3, the index feature value is 1 since it is the very first element in the (ordered) subcat sequence.

Category R:

Relation type between argument etree & predicate etree This feature is a combination of *position* and *modifying relation*. *Position* is a binary valued standard feature to describe if the argument is before or after the predicate in a parse tree. For each argument etree and intermediate etree, we consider three types of modifying relations they may have with the predicate etree: *modifying* (value 1), *modified* (value 2) and neither (value 3). From Figure 4, we can see $e1$ modifies $e0$ (predicate tree). So their modifying relation type value is 1; Combining this value with the position value, this feature for $e1$ is "1_*after*".

Attachment point of argument etree This feature describes where the argument etree is sister-adjoined/adjoined to the predicate etree, or the other way around. For $e1$ in the example, *VP* in the predicate tree is the attachment point.

Distance This feature is the number of intermediate etrees between argument etree and predicate etree in the derivation tree. In Figure 4, the distance from $e4$ to the predicate etree is 1 since only one intermediate etree $e3$ is between them.

Category I:

Intermediate etree related features *Intermediate etrees* are those etrees that are located between the predicate etree and argument etrees. The set of features we propose for each intermediate etree is quite similar to those for argument etrees except we do not consider the named-entity label for head words in this case.

Relation type of intermediate etree & predicate etree.

Attachment point of intermediate etree.

Distance between intermediate etree and predicate etree.

Up to 6 intermediate etrees are considered and the category I features are extracted for each of them (if they exist).

Each etree represents a linguistically meaningful fragment. The features of relation type, attachment point as well as the distance characterize the topological relations among the relevant etrees. In particular, the attachment point and distance features can explicitly capture important information hidden in the standard path feature. The intermediate tree related features can give richer contextual information between predicate tree and argument trees. We added the **subcat index** feature to be complementary to the *sub-cat* and *syntactic frame* features in the standard feature set.

3 Standard Feature Set

Our standard feature set is a combination of features proposed by (Gildea and Jurafsky, 2002), (Surdeanu et al., 2003; Pradhan et al., 2004; Pradhan et al., 2005b) and (Xue and Palmer, 2004). All features listed in Table 1 are used for argument classification in our baseline system; and features with asterisk are not used for argument identification[2]. We compare this baseline SRL system with a system that includes a combination of these features with the LTAG-based features. Our baseline uses *all* features that have been used in the state-of-the-art SRL systems and as our experimental results show, these standard features do indeed obtain state-of-the-art

[2]This is a standard idea in the SRL literature: removing features more useful for classification, e.g. named entity features, makes the classifier for identification more accurate.

Table 1: Standard features adopted by a typical SRL system. Features with asterisk $*$ are not used for argument identification.

Basic features from (Gildea and Jurafsky, 2002)
• **predicate lemma and voice**
• **phrase type and head word**
• **path from phrase to predicate** [1]
• **position:** phrase relative to predicate: before or after
• **sub-cat** records the immediate structure that expands from predicate's parent [2]
Additional features proposed by (Surdeanu et al. 2003; Pradhan et al., 2004, 2005)
• **predicate POS**
• **head word POS**
• **first/last word/POS**
• **POS of word immediately before/after phrase**
• **path length** [1]
• **LCA(Lowest Common Ancestor) path** from phrase to its lowest common ancestor with predicate
• **punctuation immediately before/after phrase**$*$
• **path trigrams**$*$: up to 9 are considered
• **head word named entity label** such as "PER, ORG, LOC"$*$
• **content word named entity label for PP parent node**$*$
Additional features proposed by (Xue and Palmer, 2004)
• **predicate_phrase type**
• **predicate_head word**
• **voice_position**
• **syntactic frame**$*$

[1] In Fig. 2 *NNS*$\uparrow$*NP*$\downarrow$*S*$\downarrow$*VP*$\downarrow$*VB* is the path from the constituent *NNS(agreements)* to the predicate *VB(sell)* and the path length is 4.

[2] This feature is different from the **frame** feature which usually refers to all the semantic participants for the particular predicate.

accuracy on the SRL task. We will show that adding LTAG-based features can improve the accuracy over this very strong baseline.

4 Experiments

4.1 Experimental Settings

Training data (PropBank Sections 2-21) and test data (PropBank Section 23) are taken from CoNLL-2005 shared task[3] All the necessary annotation information such as predicates, parse trees as well as Named Entity labels is part of the data. The argument set we consider is $\{$A0, A1, A2, A3, A4, AM$\}$ where AM is a generalized annotation of all adjuncts such as AM-TMP, AM-LOC, etc., where PropBank function tags like TMP or LOC in AM-TMP, AM-LOC are ignored (a common setting for SRL, see (Xue and Palmer, 2004; Moschitti, 2004)). We chose these labels for our experiments because they have sufficient training/test data for the performance comparison and provide sufficient counts for accurate significance testing. However, we also provide the evaluation result on the test set for full CoNLL-2005 task (all argument types).

We use SVM-light[4] (Joachims, 1999) with a polynomial kernel (degree=3) as our binary classifier for argument classification. We applied a linear kernel to argument identification because the training cost of this phase is extremely computationally expensive. We use 30% of the training samples to fine tune the regularization parameter c and the loss-function cost parameter j for both stages of argument identification and classification. With parameter validation experiments, we set $c = 0.258$ and $j = 1$ for the argument identification learner and $c = 0.1$ and $j = 4$ for the argument classification learner.

The classification performance is evaluated using *Precision/Recall/F-score* (p/r/f) measures. We extracted all the gold labels of A0-A4 and AM with the argument constituent index from the original test data as the "gold output". When we evaluate, we contrast the output of our system with the gold output and calculate the p/r/f for each argument type.

Our evaluation criteria which is based on predicting the SRL for *constituents* in the parse tree is based on the evaluation used in (Toutanova et al., 2005). However, we also predict and evaluate those PropBank arguments which do not have a corresponding constituent in the gold parse tree or the automatic parse tree: the *missing constituent* case. We also evaluate *discontinuous* PropBank arguments using the notation used in the CoNLL-2005 data-set but we do not predict them. This is contrast with some previous studies where the problematic cases have been usually discarded or the largest constituents in the parse tree that almost capture the missing constituent cases are picked as being the correct answer. Note that, in addition to the constituent based evalu-

[3] http://www.lsi.upc.es/∼srlconll/.

[4] http://svmlight.joachims.org/

	Gold Standard		Charniak Parser	
	std	std+ltag	std	std+ltag
p(%)	95.66	96.79	87.71	89.11
r(%)	94.36	94.59	84.86	85.51
f(%)	95.00	**95.68**	86.26	**87.27**$*$

Table 2: Argument identification results on test data

ation, in Section 4.4 we also provide the evaluation of our model on the CoNLL-2005 data-set.

Because the main focus of this work is to evaluate the impact of the LTAG-based features, we did not consider the frameset or a distribution over the entire argument set or apply any inference/constraints as a post-processing stage as most current SRL systems do. We focus our experiments on showing the value added by introducing LTAG-based features to the SRL task over and above what is currently used in SRL research.

4.2 Argument Identification

Table 2 shows results on argument identification (a binary classification of constituents into argument or non-argument). To fully evaluate the influence of the LTAG-based features, we report the identification results on both Gold Standard parses and on Charniak parser output (Charniak, 2000)[5].

As we can see, after combing the LTAG-based features with the standard features, F-score increased from 95.00% to 95.68% with Gold-standard parses; and from 86.26% to 87.27% with the Charniak parses (a larger increase). We can see LTAG-based features help in argument identification for both cases. This result is better than (Xue and Palmer, 2004), and better on gold parses compared to (Toutanova et al., 2005; Punyakanok et al., 2005b).

4.3 Argument Classification

Based on the identification results, argument classification will assign the semantic roles to the argument candidates. For each argument of A0-A4 and AM, a "one-vs-all" SVM classifier is trained on both the standard feature set (std) and the augmented feature set (std+ltag). Table 3 shows the classification results on the Gold-standard parses with the gold argument identification; Table 4 and 5 show the classification results on the Charniak parser with the gold argument identification and the automatic argument identification respectively. Scores for multi-class SRL are calculated based on the total number of correctly predicted labels, total number of gold labels and the number of labels in our prediction for this argument set.

[5]We use the parses supplied with the CoNLL-2005 shared task for reasons of comparison.

class	std(p/r/f)%		std+ltag(p/r/f)%	
A0	96.69	96.71	96.71	96.77
	96.70		**96.74**	
A1	93.82	93.30	97.30	94.87
	93.56		**96.07**	
A2	87.05	79.98	92.43	81.42
	83.37		**86.58**	
A3	94.44	68.79	97.69	73.41
	79.60		**83.33**	
A4	96.55	82.35	94.11	78.43
	88.89		85.56	
AM	98.41	96.61	98.67	97.88
	97.50		**98.27**	
multi-class	95.35	93.62	97.15	94.70
	94.48		**95.91**	

Table 3: Argument classification results on Gold-standard parses with gold argument boundaries

4.4 Discussion

From the results shown in the tables, we can see that by adding the LTAG-based features, the overall performance of the systems is improved both for argument identification and for argument classification.

Table 3 and 4 show that with the gold argument identification, the classification for each class in {A0, A1, A2, A3, AM} consistently benefit from LTAG-based features. Especially for A3, LTAG-based features lead to more than 3 percent improvement. But for A4 arguments, the performance drops 3 percent in both cases. As we noticed in Table 5, which presents the argument classification results on Charniak parser output with the automatic argument identification, the prediction accuracy for classes A0, A1, A3, A4 and AM is improved, but drops a little for A2.

In addition, we also evaluated our feature set on the full CoNLL 2005 shared task. The over-

class	std(p/r/f)%		std+ltag(p/r/f)%	
A0	96.04	92.92	96.07	92.92
	94.46		**94.47**	
A1	90.64	85.71	94.64	86.67
	88.11		**90.48**	
A2	84.46	75.72	89.26	75.22
	79.85		**81.64**	
A3	87.50	62.02	87.10	68.35
	72.59		**76.60**	
A4	90.00	79.12	90.54	73.62
	84.21		81.21	
AM	95.14	85.54	96.60	86.51
	90.09		**91.27**	
multi-class	93.25	86.45	94.71	87.15
	89.72		**90.77**	

Table 4: Argument classification results on Charniak parser output with gold argument boundaries

class	std(p/r/f)%		std+ltag(p/r/f)%	
A0	86.50	86.18	88.17	87.70
	86.34		**87.93**∗	
A1	78.73	83.82	88.78	85.22
	81.19		**86.97**∗	
A2	85.40	73.93	83.11	75.42
	79.25		79.08	
A3	85.71	60.76	85.71	68.35
	71.11		**76.06**∗	
A4	84.52	78.02	89.47	74.72
	81.15		**81.43**	
AM	80.47	82.11	83.87	81.54
	81.29		**82.69**∗	
multi-class	81.79	82.90	86.04	84.47
	82.34		**85.25**∗	

Table 5: Argument classification results on Charniak parser output with automatic argument boundaries

all performance using LTAG features increased from 74.41% to 75.31% in terms of F-score on the full argument set. Our accuracy is most closely comparable to the 78.63% accuracy achieved on the full task by (Pradhan et al., 2005a). However, (Pradhan et al., 2005a) uses some additional information since it deals with incorrect parser output by using multiple parsers. The 79.44% accuracy obtained by the top system in CoNLL 2005 (Punyakanok et al., 2005a) is not directly comparable since their system used the more accurate n-best parser output of (Charniak and Johnson, 2005). In addition their system also used global inference. Our focus in this paper was to propose *new LTAG features* and to evaluate impact of these features on the SRL task.

We also compared our proposed feature set against predicate/argument features (PAF) proposed by (Moschitti, 2004). We conducted an experiment using *SVM-light-TK-1.2 toolkit*[6]. The PAF tree kernel is combined with the standard feature vectors by a linear operator. With settings of Table 5, its multi-class performance (p/r/f)% is 83.09/80.18/81.61 with linear kernel and 85.36/81.79/83.53 with polynomial kernel (degree=3) over *std* feature vectors.

4.5 Significance Testing

To assess the statistical significance of the improvements in accuracy we did a two-tailed significance test on the results of both Table 2 and 5 where Charniak's parser outputs were used. We chose *SIGF*[7], which is an implementation of a computer-intensive, stratified approximate-randomization test (Yeh, 2000). The statistical difference is assessed on SRL identification, classification for each class (A0-A4, AM) and the full SRL task (overall performance). In Table 2 and 5, we labeled numbers under *std+ltag* that are statistically significantly better from those under *std* with asterisk. The significance tests show that for identification and full SRL task, the improvements are statistically significant with p value of 0.013 and 0.0001 at a confidence level of 95%. The significance test on each class shows that the improvement by adding LTAG-based features is statistically significant for class A0, A1, A3 and AM. Even though in Table 5 the performance of A2 appears to be worse it is not significantly so, and A4 is not significantly better. In comparison, the performance of PAF did not show significantly better than *std* with p value of 0.593 at the same confidence level of 95%.

[6]http://ai-nlp.info.uniroma2.it/moschitti/TK1.2-software/Tree-Kernel.htm

[7]http://www.coli.uni-saarland.de/∼pado/sigf/index.html

	full	**full−P**	**full−R**	**full−S**	**full−A**	**full−I**	**std**
id	90.5	90.6	90.0	90.5	90.5	90.1	89.6
A0	84.5	84.3	84.6	84.5	84.3	83.5	84.2
A1	89.8	90.1	89.4	89.3	89.6	89.3	88.9
A2	84.2	84.2	84.0	83.7	83.6	83.6	84.9
A3	76.7	80.7	75.1	76.0	75.6	76.7	78.6
A4	80.0	83.3	80.0	79.6	80.0	80.0	79.2
AM	82.8	83.3	82.9	82.8	82.6	83.1	82.4

Table 6: Impact of each LTAG feature category (**P, R, S, A, I** defined in Section 2.1.3) on argument classification and identification on CoNLL-2005 development set (WSJ Section 24). **full** denotes the full feature set, and we use $-\alpha$ to denote removal of a feature category of type α. For example, **full−P** is the feature set obtained by removing all **P** category features. **std** denotes the standard feature set.

5 Analysis of the LTAG-based features

We analyzed the drop in performance when a particular type of LTAG feature category is removed from the full set of LTAG features (we use the broad categories **P, R, S, A, I** as defined in Section 2.1.3). Table 6 shows how much performance is lost (or gained) when a particular type of LTAG feature is dropped from the full set.

These experiments were done on the development set from CoNLL-2005 shared task, using the provided Charniak parses. All the SVM models were trained using a polynomial kernel with degree 3. It is clear that the **S, A, I** category features help in most cases and **P** category features hurt in most cases, including argument identification. It is also worth noting that the **R** and **I** category features help most for identification. This vindicates the use of LTAG derivations as a way to generalize long paths in the parse tree between the predicate and argument. Although it seems LTAG features have negative impact on prediction of A3 arguments on this development set, dropping the **P** category features can actually improve performance over the standard feature set. In contrast, for the prediction of A2 arguments, none of the LTAG feature categories seem to help.

Note that since we use a polynomial kernel in the full set, we cannot rule out the possibility that a feature that improves performance when dropped may still be helpful when combined in a non-linear kernel with features from other categories. However, this analysis on the development set does indicate that overall performance may be improved by dropping the **P** feature category. We plan to examine this effect in future work.

6 Related Work

There has been some previous work in SRL that uses LTAG-based decomposition of the parse tree. (Chen and Rambow, 2003) use LTAG-based decomposition of parse trees (as is typically done for statistical LTAG parsing) for SRL. Instead of extracting a typical "standard" path feature from the derived tree, (Chen and Rambow, 2003) uses the path within the elementary tree from the predicate to the constituent argument. Under this frame, they only recover semantic roles for those constituents that are localized within a single etree for the predicate, ignoring cases that occur outside the etree. As stated in their paper, "as a consequence, adjunct semantic roles (ARGM's) are basically absent from our test corpus"; and around 13% complement semantic roles cannot be found in etrees in the gold parses. In contrast, we recover all SRLs by exploiting more general paths in the LTAG derivation tree. A similar drawback can be found in (Gildea and Hockenmaier, 2003) where a **parse tree path** was defined in terms of Combinatory Categorial Grammar (CCG) types using grammatical relations between predicate and arguments. The two relations they defined can only capture 77% arguments in Propbank and they had to use a standard path feature as a replacement when the defined relations cannot be found in CCG derivation trees. In our framework, we use intermediate sub-structures from LTAG derivations to capture these relations instead of bypassing this issue.

Compared to (Liu and Sarkar, 2006), we have used a more sophisticated learning algorithm and a richer set of syntactic LTAG-based features in this task. In particular, in this paper we built a strong baseline system using a standard set of features and did a thorough comparison between this strong baseline and our proposed system with LTAG-based features. The experiments in (Liu and Sarkar, 2006) were conducted on gold parses and it failed to show any improvements after adding LTAG-based features. Our experimental results show that LTAG-based features can help improve the performance of SRL systems. While (Liu and Sarkar, 2006) propose some new features for SRL based on LTAG derivations, we propose several novel features and in addition they do not show that their features are useful for SRL.

Our approach shares similar motivations with the approach in (Shen and Joshi, 2005) which uses PropBank information to recover an LTAG treebank as if it were hidden data underlying the Penn Treebank. However their goal was to extract an LTAG grammar using PropBank information from the Treebank, and *not* the SRL task.

Features extracted from LTAG derivations are different and provide distinct information when compared to predicate-argument features (PAF) or subcategorization features (SCF) used in (Moschitti, 2004) or even the later use of argument spanning trees (AST) in the same framework. The *adjunction* operation of LTAG and the extended domain of locality is not captured by those features as we have explained in detail in Section 2.

7 Conclusion and Future Work

In this paper we show that LTAG-based features improve on the best known set of features used in current SRL prediction systems: the F-score for argument identification increased from 86.26% to 87.27% and from 82.34% to 85.25% for the SRL task. The analysis of the impact of each LTAG feature category shows that the intermediate etrees are important for the improvement. In future work we plan to explore the impact that different types of LTAG derivation trees have on this SRL task, and explore the use of tree kernels defined over the LTAG derivation tree. LTAG derivation tree kernels were previously used for parse re-ranking by (Shen et al., 2003). Our work also provides motivation to do SRL and LTAG parsing simultaneously.

Acknowledgements

This research was partially supported by NSERC, Canada (RGPIN: 264905). We would like to thank Aravind Joshi, Libin Shen, and the anonymous reviewers for their comments.

References

E. Charniak and M. Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *ACL-2005*.

E. Charniak. 2000. A maximum-entropy-inspired parser. In *NAACL-2000*.

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

J. Chen and K. Vijay-Shanker. 2000. Automated Extraction of TAGs from the Penn Treebank. In *Proc. of the 6th International Workshop on Parsing Technologies (IWPT-2000), Italy*.

D. Chiang. 2000. Statistical parsing with an automatically extracted tree adjoining grammars. In *ACL-2000*.

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

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*.

A. Joshi and Y. Schabes. 1997. Tree-adjoining grammars. *Handbook of Formal Languages*, 3.

Y. Liu and A. Sarkar. 2006. Using LTAG-Based Features for Semantic Role Labeling. In *TAG+8-2006*.

R. McDonald, K. Crammer, and F. Pereira. 2005. Online Large-Margin Training of Dependency Parsers. In *ACL-2005*.

A. Moschitti. 2004. A study on convolution kernels for shallow semantic parsing. In *ACL-2004*.

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, K. Hacioglu, W. Ward, J. Martin, and D. Jurafsky. 2005a. Semantic role chunking combining complementary syntactic views. In *CoNLL-2005 Shared Task*.

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

V. Punyakanok, D. Roth, and W. Yih. 2005a. Generalized inference with multiple semantic role labeling systems (shared task paper). In *CoNLL-2005*.

V. Punyakanok, D. Roth, and W. Yih. 2005b. The necessity of syntactic parsing for semantic role labeling. In *IJCAI-2005*.

L. Shen and A. Joshi. 2005. Building an LTAG Treebank. Technical Report Technical Report MS-CIS-05-15,5, CIS Department, University of Pennsylvania.

L. Shen, A. Sarkar, and A. Joshi. 2003. Using LTAG based features in parse reranking. In *EMNLP-2003*.

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

K. Toutanova, A. Haghighi, and C. D. Manning. 2005. Joint learning improves semantic role labeling. In *ACL-2005*.

F. Xia. 1999. Extracting tree adjoining grammars from bracketed corpora. In *Proceedings of 5th Natural Language Processing Pacific Rim Symposium (NLPRS-99)*, Beijing, China.

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

A. Yeh. 2000. More accurate tests for the statistical significance of result differences. In *COLING-2000*.