Dependency Tree-based SRL with Proper Pruning and Extensive Feature Engineering

Hongling Wang Honglin Wang Guodong Zhou Qiaoming Zhu
JiangSu Provincial Key Lab for Computer Information Processing Technology
School of Computer Science and Technology,
Soochow University, Suzhou, China 215006
{redleaf, 064227065055,gdzhou, qmzhu}@suda.edu.cn

Abstract

This paper proposes a dependency tree-based SRL system with proper pruning and extensive feature engineering. Official evaluation on the CoNLL 2008 shared task shows that our system achieves 76.19 in labeled macro F1 for the overall task, 84.56 in labeled attachment score for syntactic dependencies, and 67.12 in labeled F1 for semantic dependencies on combined test set, using the standalone MaltParser. Besides, this paper also presents our unofficial system by 1) applying a new effective pruning algorithm; 2) including additional features; and 3) adopting a better dependency parser, MSTParser. Unofficial evaluation on the shared task shows that our system achieves 82.53 in labeled macro F1, 86.39 in labeled attachment score, and 78.64 in labeled F1, using MSTParser on combined test set. This suggests that proper pruning and extensive feature engineering contributes much in dependency tree-based SRL.

1 Introduction

Although CoNLL 2008 shared task mainly evaluates joint learning of syntactic and semantic parsing, we focus on dependency tree-based semantic role labeling (SRL). SRL refers to label the semantic roles of predicates (either verbs or nouns) in a sentence. Most of previous SRL systems (Gildea and Jurafsky, 2002; Gildea and Palmer, 2002; Punyakanok et al., 2005; Pradhan et al., 2004, 2005) work on constituent structure trees and has shown to achieve remarkable results. For example, Punyakanok et al. (2005) achieved the best performance in the CoNLL 2005 shared task with 79.44 in F-measure on the WSJ test set and 77.92 on the combined test set (WSJ +Brown).

With rapid development of dependency parsing in the last few years, more and more researchers turn to dependency tree-based SRL with hope to advance SRL from viewpoint of dependency parsing. Hacioglu (2004) pioneered this work by formulating SRL as a classification problem of mapping various dependency relations into semantic roles. Compared with previous researches on constituent structure tree-based SRL which adopts constituents as labeling units, dependency tree-based SRL adopts dependency relations as labeling units. Due to the difference between constituent structure trees and dependency trees, their feature spaces are expected to be somewhat different.

In the CoNLL 2008 shared task, we extend the framework by Hacioglu (2004) with maximum entropy as our classifier. For evaluation, we will mainly report our official SRL performance using MaltParser (Nivre and Nilsson, 2005). Besides, we will also present our unofficial system by 1) applying a new effective pruning algorithm; 2) including additional features; and 3) adopting a better dependency parser, MSTParser (McDonald, 2005).

In the remainder of this paper, we will briefly describe our system architecture, present various features used by our models and report the performance on CoNLL 2008 shared task (both official and unofficial).

2 System Description

In CoNLL 2008 shared task, we adopt a standard three-stage process for SRL: pruning, argument identification and argument classification. To model the difference between verb and noun predicates, we carry out separate training and testing for verb and noun predicates respectively.

In addition, we adopt OpenNLP maximum entropy package[1] in argument identification and classification.

2.1 Predicate identification

Most of Previous SRL systems only consider given predicates. However, predicates are not given in CoNLL 2008 shared task and required to be determined automatically by the system. Therefore, the first step of the shared task is to identify the verb and noun predicates in a sentence. Due to time limitation, a simple algorithm is developed to identify noun and verb predicates:

1) For the WSJ corpus, we simply adopt the annotations provided by PropBank and NomBank. That is, we only consider the verb and noun predicates annotated in PropBank and NomBank respectively.
2) For the Brown corpus, verb predicates are identified simply according to its POS tag and noun predicates are determined using a simple method that only those nouns which can also be used as verbs are identified. To achieve this goal, an English lexicon of about 56K word is applied to identify noun predicates.

Evaluation on the test set of CoNLL 2008 shared task shows that our simple predicate identification algorithm achieves the accuracies of 98.6% and 92.7 in the WSJ corpus for verb and noun predicates respectively, with overall accuracy of 95.5%, while it achieves the accuracies of 73.5% and 43.1% in the Brown corpus for verb and noun predicates respectively with overall accuracy of 61.8%. This means that the performance of predicate identification in the Brown corpus is much lower than the one in the WSJ corpus. This further suggests that much work is required to achieve reasonable predicate identification performance in future work.

2.2 Preprocessing

Using the dependency relations returned by a dependency parser (either MaltParser or MSTParser in this paper), we can construct corresponding dependency tree for a given sentence. For example, Figure 1 shows the dependency tree of the sentence "Meanwhile, overall evidence on the economy remains fairly clouded.". Here, W is composed of two parts: word and its POS tag with "/" as a separator while R means a dependency relation and ARG represents a semantic role.

In Hacioglu (2004), a simple pruning algorithm is applied to filter out unlikely dependency relation nodes in a dependency tree by only keeping the parent/children/grand-children of the predicate, the siblings of the predicates, and the children/grandchildren of the siblings. This paper extends the algorithm a little bit by including the nodes two more layers upward and downward with regard to the predicate's parent, such as the predicate's grandparent, the grandparent's children and the grandchildren's children. For the example as shown in Figure 1, all the nodes in the entire tree are kept. Evaluation on the training set shows that our pruning algorithm significantly reduces the training instances by 76.9%. This is at expanse of wrongly pruning 1.0% semantic arguments for verb predicates. However, this figure increases to 43.5% for noun predicates due to our later observation that about half of semantic arguments of noun predicates distributes over ancestor nodes out of our consideration. This suggests that a specific pruning algorithm is necessary for noun predicates to include more ancestor nodes.

2.3 Features

Some of the features are borrowed from Hacioglu (2004) with some additional features motivated by constituent structure tree-based SRL (Pradhan et al 2005; Xue and Palmer, 2004). In the following, we explain these features and give examples with regard to the dependency tree as shown in Figure 1. We take the word *evidence* in Figure 1 as the predicate and the node "*on*" as the node on focus.

The following eight basic features are motivated from constituent structure tree-based SRL:

1) **Predicate**: predicate lemma. (evidence)
2) **Predicate POS**: POS of current predicate. (NN)
3) **Predicate Voice**: Whether the predicate (verb) is realized as an active or passive construction. If the predicate is a noun, the value is null and presented as "_". (_)

[1]https://sourceforge.net/project/showfiles.php?group_id=59
61

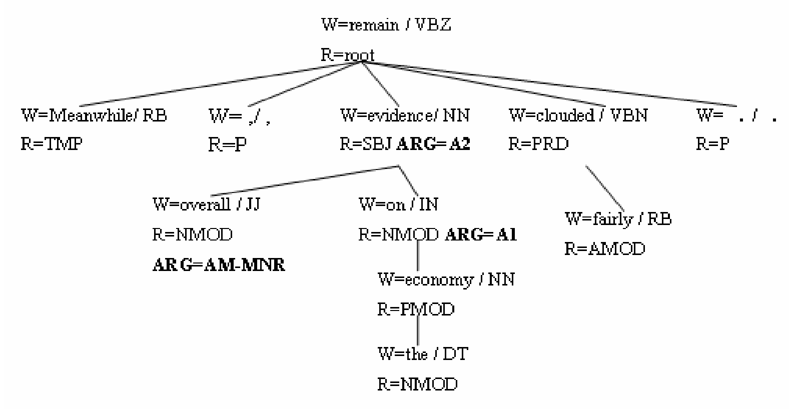

Figure 1. Example of a dependency tree augmented with semantic roles for the given predicate *evidence*.

4) **Relation type**: the dependency relation type of the current node. (NMOD)

5) **Path**: the chain of relations from current relation node to the predicate. (NMOD->SBJ)

6) **Sub-categorization**: The relation type of predicate and the left-to-right chain of the relation label sequence of the predicate's children. (SBJ->NMOD-NMOD)

7) **Head word**: the head word in the relation, that is, the headword of the parent of the current node. (evidence)

8) **Position**: the position of the headword of the current node with respect to the predicate position in the sentence, which can be before, after or equal. (equal)

Besides, we also include following additional features borrowed from Hacioglu (2004):

1) **Family membership**: the relationship between current node and the predicate node in the family tree, such as parent, child, sibling. (child)

2) **Dependent word**: the modifying word in the relation, that is, the word of current node. (on)

3) **POS of headword**: the POS tag of the headword of current word. (NN)

4) **POS of dependent word**: the POS tag of current word. (IN)

5) **POS pattern of predicate's children**: the left-to-right chain of the POS tag sequence of the predicate's children. (JJ-IN)

6) **Relation pattern of predicate's children**: the left-to-right chain of the relation label sequence of the predicate's children. (NMOD-NMOD)

7) **POS pattern of predicate's siblings**: the left-to-right chain of the POS tag sequence of the predicate's siblings. (RB-.-VBN-.)

8) **Relation pattern of predicate's siblings**: the left-to-right chain of the relation label sequence of the predicate's siblings. (TMP-P-PRD-P)

3 System Performance

All the training data are included in our system, which costs 70 minutes in training and 5 seconds on testing on a PC platform with a Pentium D 3.0G CPU and 2G Memory. In particular, the argument identification stage filters out those nodes whose probabilities of not being semantic arguments are more than 0.98 for verb and noun predicates.

	Labeled Macro F1	Labeled F1	LAS
Test WSJ	78.39	70.41	85.50
Test Brown	59.89	42.67	77.06
Test WSJ+Brown	**76.19**	**67.12**	**84.56**

Table 1: Official performance using MaltParser (with the SRL model trained and tested on the automatic output of MaltParser)

All the performance is returned on the test set using the CoNLL 2008 evaluation script *eval08.pl* provided by the organizers. Table 1 shows the official performance using MaltParser (with the SRL model trained and tested on the automatic output of MaltParser provided by the task organizers) as the dependency parser. It shows that our system performs well on the WSJ corpus and badly on the Brown corpus largely due to bad performance on predicate identification.

4 Post-evaluation System

To gain more insights into dependency tree-based SRL, we improve the system with a new pruning algorithm and additional features, after submitting our official results.

4.1 Effective pruning

Our new pruning algorithm is motivated by the one proposed by Xue and Palmer (2004), which only keeps those siblings to a node on the path from current predicate to the root are included, for constituent structure tree-based SRL. Our pruning algorithm further cuts off the nodes which are not related with the predicate. Besides, it filters out those nodes which are punctuations or with "symbol" dependency relations. Evaluation on the Brown corpus shows that our pruning algorithm significantly reduces the training data by 75.5% at the expense of wrongly filtering out 0.7% and 0.5% semantic arguments for verb and noun predicates respectively. This suggests that our new pruning algorithm significantly performs better than the old one in our official system, especially for the identification of noun predicates.

Furthermore, the argument identification stage filters out those nodes whose probabilities of not being semantic arguments are more than 0.90 and 0.85 for verb and noun predicates respectively, since we that our original threshold of 0.98 in the official system is too reserved.

Finally, those rarely-occurred semantic roles which occur less than 200 in the training set are filtered out and thus not considered in our system, such as A5, AA, C-A0, C-AM-ADV, R-A2 and SU.

4.2 Extensive Feature Engineering

Motivated by constituent structure tree-based SRL, two more combined features are considered in our post-evaluation system:

1) **Predicate + Headword**: (evidence + remain)
2) **Headword + Relation**: (remain + Root)

In order to better evaluate the contribution of various additional feature, we build a baseline system using hand-corrected dependency relations and the eight basic features, motivated by constituent structure tree-based SRL, as described in Section 2.3. Table 2 shows the effect of various additional features by adding one individually to the baseline system. It shows that the feature of dependent word is most useful, which improves the labeled F1 score from 81.38% to 84.84%. It also shows that the two features about predicate's sibling deteriorate the performance. Therefore, we delete these two features from remaining experiments. Although the combined feature of "predicate+head word" is useful in constituent structure tree-based SRL, it slightly decrease the performance in dependency tree-based SRL. For convenience, we include it in our system.

	P	R	F1
Baseline	84.31	78.64	81.38
+ Family membership	84.70	78.87	81.68
+ Dependent word	86.74	83.01	84.84
+ POS of headword	84.44	78.55	81.38
+ POS of dependent word	84.42	78.33	81.47
+ POS pattern of predicate's children	84.35	78.73	81.47
+ Relation pattern of predicate's children	84.75	78.97	81.76
+ Relation pattern of predicate's siblings	84.29	78.52	81.30
+ POS pattern of predicate's siblings	83.75	78.32	80.95
+ Predicate + Headword	83.30	78.94	81.30
+Headword + Relation	84.66	79.37	81.93

Table 2: Effects of various additional features

4.3 Best performance

Table 3 shows our system performance after applying above effective pruning strategy and additional features using the default MaltParser. Table 3 also reports our performance using the state-of-the-art MSTParser. To show the impact of predicate identification in dependency tree-based SRL, Table 4 report the performance on gold predicate identification, i.e. only using annotated predicates in the corpora.

Comparison of Table 1 and Table 3 using the MaltParser shows that our new extension with effective pruning and extensive engineering significantly improves the performance. It also shows that MSTParser-based SRL performs slightly better than MaltParser-based one, much less than the performance difference on dependency parsing between them. This suggests that such difference between these two state-of-the-art dependency parsers does not much affect corresponding SRL systems. This is also confirmed by the results in Table 4.

Comparison of Table 3 and Table 4 in labeled F1 on the Brown test data shows that the system with gold predicate identification significantly outperforms the one with automatic predicate identification using our simple algorithm by about 22 in labeled F1. This suggests that the performance of predicate identification is critical to SRL.

	MSTParser			MaltParser		
	Labeled Macro F1	Labeled F1	LAS	Labeled Macro F1	Labeled F1	LAS
Test WSJ	84.50	**81.95**	87.01	83.69	**81.82**	85.50
Test Brown	67.61	**53.69**	81.46	65.09	**53.03**	77.06
Test WSJ+Brown	82.53	**78.64**	86.39	81.52	**78.45**	84.56

Table 3: Unofficial performance using MSTParser and MaltParser with predicates automatically identified

	MSTParser			MaltParser		
	Labeled Macro F1	Labeled F1	LAS	Labeled Macro F1	Labeled F1	LAS
Test WSJ	84.75	**82.45**	87.01	84.04	**82.52**	85.50
Test Brown	78.31	**75.07**	81.46	75.72	**74.28**	77.06
Test WSJ+Brown	84.05	**81.66**	86.39	83.13	**81.64**	84.56

Table 4: Unofficial performance using MSTParser and MaltParser with gold predicate identification

5 Conclusions

This paper presents a dependency tree-based SRL system by proper pruning and extensive feature engineering. Evaluation on the CoNLL 2008 shared task shows that proper pruning and extensive feature engineering contributes much. It also shows that SRL heavily depends on the performance of predicate identification.

In future work, we will explore better ways in predicate identification. In addition, we will explore more on dependency parsing and further joint learning on syntactic and semantic parsing.

Acknowledgment

This research is supported by Project 60673041 under the National Natural Science Foundation of China and Project 2006AA01Z147 under the “863” National High-Tech Research and Development of China.

References

Gildea, Daniel and Daniel Jurafsky. 2002. Automatic Labeling of Semantic Roles. *Computational Linguistics*, 28:3, pages 245-288.

Gildea, Daniel and Martha Palmer. 2002. The Necessity of Syntactic Parsing for Predicate Argument Recognition. *In Proceedings of the 40th Association for Computational Linguistics*, 2002.

Hacioglu, Kadri. 2004. Semantic Role Labeling Using Dependency Trees. *In Proceedings of the International Conference on Computational Linguistics (COLING).* 2004.

McDonald, Ryan, Fernando Pereira, Kiril Ribarov, Jan Hajiˇ. 2005. Non-Projective Dependency Parsing using Spanning Tree Algorithms. *In the proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, 2005

Surdeanu, Mihai, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 Shared Task on Joint Parsing of Syntactic and Semantic Dependencies. In *Proceedings of the 12th Conference on Computational Natural Language Learning (CoNLL-2008).*

Nivre, Joakim and Jens Nilsson. 2005. Pseudo-Projective Dependency Parsing. *In Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pp. 99-106, 2005

Pradhan, Sameer, Wayne Ward, Kadri Hacioglu, James H. Martin, Dan Jurafsky. 2004. Shallow Semantic Parsing Using Support Vector Machines. *In Proceedings of (HLT-NAACL-2004)*, 2004.

Pradhan, Sameer, Wayne Ward, Kadri Hacioglu, James H. Martin, Dan Jurafsky. 2005. Semantic role labeling using different syntactic views. *In Proceedings of the 43rd Association for Computational Linguistics (ACL-2005)*, 2005.

Punyakanok, Vasin, Peter Koomen, Dan Roth, and Wen-tau Yih. 2005. Generalized inference with multiple semantic role labeling systems. *In Proceedings of 9th Conference on Computational Natural Language Learning (CoNLL-2005).*2005

Xue, Nianwen and Martha Palmer. 2004. Calibrating features for semantic role labeling. *In Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2004.