Maximum Entropy based Semantic Role Labeling

Kyung-Mi Park and **Hae-Chang Rim**
Department of Computer Science & Engineering, Korea University
5-ka, Anam-dong, SeongBuk-gu, SEOUL, 136-701, KOREA
{kmpark, rim}@nlp.korea.ac.kr

1 Introduction

The semantic role labeling (SRL) refers to finding the semantic relation (e.g. *Agent*, *Patient*, etc.) between a predicate and syntactic constituents in the sentences. Especially, with the argument information of the predicate, we can derive the predicate-argument structures, which are useful for the applications such as automatic information extraction. As previous work on the SRL, there have been many machine learning approaches. (Gildea and Jurafsky, 2002; Pradhan et al., 2003; Lim et al., 2004).

In this paper, we present a two-phase SRL method based on a maximum entropy (ME) model. We first identify parse constituents that represent valid semantic arguments of a given predicate, and then assign appropriate semantic roles to the the identified parse constituents. In the two-phase SRL method, the performance of the argument identification phase is very important, because the argument classification is performed on the region identified at the identification phase. In this study, in order to improve the performance of identification, we try to incorporate *clause boundary* restriction and *tree distance* restriction into pre-processing of the identification phase.

Since features for identifying arguments are different from features for classifying a role, we need to determine different feature sets appropriate for the tasks. We determine final feature sets for each phase with experiments. We participate in the closed challenge of the CoNLL-2005 shared task and report results on both development and test sets. A detailed description of the task, data and related work can be found in Carreras and Màrquez (2005).

2 System Description

In this section, we describe our system that identifies and classifies semantic arguments. First, we explain pre-processing of the identification phase. Next, we describe features employed. Finally, we explain classifiers used in each phase.

2.1 Pre-processing

We thought that the occurrence of most semantic arguments are sensitive to the boundary of the immediate clause or the upper clauses of a predicate. Also, we assumed that they exist in the uniform distance on the parse tree from the predicate's parent node (called P_p) to the parse constituent's parent node (called P_c). Therefore, for identifying semantic arguments, we do not need to examine all parse constituents in a parse tree. In this study, we use the *clause boundary* restriction and the *tree distance* restriction, and they can provide useful information for spotting the probable search space which include semantic arguments.

In Figure 1 and Table 1, we show an example of applying the *tree distance* restriction. We show the distance between P_p=VP and the nonterminals of a parse tree in Figure 1. For example, NP_2:*d=3* means 3 times downward movement through the parse tree from P_p=VP to P_c=NP_2. NP_4 does not have the distance from P_p because we allow to move only upward or only downward through the tree from P_p to P_c. In Table 1, we indicate all 14 argument candidates that correspond to *tree distance* restriction (d≤3). Only 2 of the 14 argument candidates are actually served to semantic arguments (NP_4, PP).

Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL),
pages 209–212, Ann Arbor, June 2005. ©2005 Association for Computational Linguistics

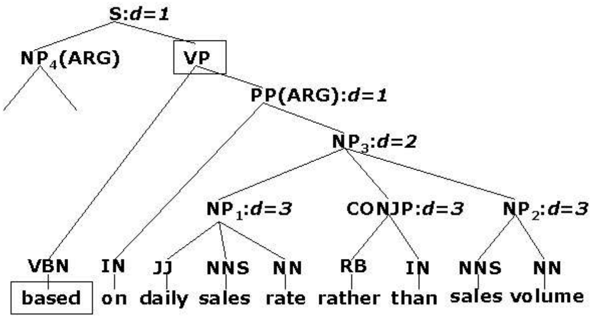

Figure 1: Distance between P$_p$=VP and P$_c$.

distance	direction	P$_c$	argument candidates
d=1	UP	S	NP$_4$
d=0	-	VP	PP
d=1	DOWN	PP	IN, NP$_3$
d=2	DOWN	NP$_3$	NP$_1$, CONJP, NP$_2$
d=3	DOWN	NP$_1$	JJ, NNS, NN
d=3	DOWN	CONJP	RB, IN
d=3	DOWN	NP$_2$	NNS, NN

Table 1: Probable argument candidates (d≤3).

2.2 Features

The following features describe properties of the verb predicate. These featues are shared by all the parse constituents in the tree.

- *pred_lex*: this is the predicate itself.
- *pred_POS*: this is POS of the predicate.
- *pred_phr*: this is the syntactic category of P$_p$.
- *pred_type*: this represents the predicate usage such as to-infinitive form, the verb predicate of a main clause, and otherwise.
- *voice*: this is a binary feature identifying whether the predicate is active or passive.
- *sub_cat*: this is the phrase structure rule expanding the predicate's parent node in the tree.
- *pt+pl*: this is a conjoined feature of *pred_type* and *pred_lex*. Because the maximum entropy model assumes the independence of features, we need to conjoin the coherent features.

The following features characterize the internal structure of a argument candidate. These features change with the constituent under consideration.

- *head_lex*: this is the headword of the argument candidate. We extracts the headword by using the Collins's headword rules.
- *head_POS*: this is POS of the headword.
- *head_phr*: this is the syntactic category of P$_c$.
- *cont_lex*: this is the content word of the argument candidate. We extracts the content word by using the *head table* of the chunklink.pl [1].
- *cont_POS*: this is POS of the content word.
- *gov*: this is the *governing category* introduced by Gildea and Jurafsky (2002).

The following features capture the relations between the verb predicate and the constituent.

- *path*: this is the syntactic path through the parse tree from the parse constituent to the predicate.
- *pos*: this is a binary feature identifying whether the constituent is before or after the predicate.
- *pos+clau*: this, conjoined with *pos*, indicates whether the constituent is located in the immediate clause, in the first upper clause, in the second upper clause, or in the third upper clause.
- *pos+VP*, *pos+NP*, *pos+SBAR*: these are numeric features representing the number of the specific chunk types between the constituent and the predicate.
- *pos+CC*, *pos+comma*, *pos+colon*, *pos+quote*: these are numeric features representing the number of the specific POS types between the constituent and the predicate .
- *pl+hl* (*pred_lex* + *head_lex*), *pl+cl* (*pred_lex* + *cont_lex*), *v+gov* (*voice* + *gov*).

2.3 Classifier

The ME classifier for the identification phase classifies each parse constituent into one of the following classes: *ARG* class or *NON-ARG* class. The ME classifier for the classification phase classifies the identified argument into one of the pre-defined semantic roles (e.g. *A0*, *A1*, *AM-ADV*, *AM-CAU*, etc.).

[1] http://pi0657.kub.nl/šabine/chunklink/chunklink_2-2-2000_for_conll.pl

	#exa.	%can.	#can.	%arg.	F$_{\beta=1}$
no restriction					
All$_1$	3,709,080	-	233,394	96.06	79.37
All$_2$	2,579,278	-	233,004	95.90	79.52
All$_3$	1,598,726	100.00	231,120	95.13	79.92
restriction on clause boundary					
1/0	1,303,596	81.54	222,238	91.47	78.97
1/1	1,370,760	85.74	223,571	92.02	79.14
2/0	1,403,630	87.80	228,891	94.21	79.66
2/1	1,470,794	92.00	230,224	94.76	79.89
3/0	1,439,755	90.06	229,548	94.48	79.63
3/1	1,506,919	94.26	230,881	95.03	79.79
restriction on tree distance					
6/1	804,413	50.32	226,875	93.38	80.17
6/2	936,021	58.55	227,637	93.69	79.94
7/1	**842,453**	**52.70**	**228,129**	**93.90**	**80.44**
7/2	974,061	60.93	228,891	94.21	80.03
8/1	871,541	54.51	228,795	94.17	80.24
8/2	1,003,149	62.75	229,557	94.48	80.04
restriction on clause boundary & tree distance					
2/1,7/1	786,951	49.22	227,523	93.65	80.12
2/1,8/1	803,040	50.23	228,081	93.88	80.11
3/1,7/1	800,740	50.09	227,947	93.82	80.28
3/1,8/1	822,225	51.43	228,599	94.09	80.06

Table 2: Different ways of reducing candidates.

	Prec.	Recall	F$_{\beta=1}$	Accu.
All	82.57	78.41	80.44	86.00
All-(pred_lex)	82.80	77.78	80.21	84.93
All-(pred_POS)	83.40	76.72	79.92	85.95
All-(pred_phr)	83.11	77.57	80.24	85.87
All-(pred_type)	82.76	77.91	80.26	**85.99**
All-(voice)	82.87	77.88	80.30	85.88
All-(sub_cat)	82.48	77.68	80.00	84.88
All-(pt+pl)	83.20	77.40	80.20	85.62
All-(head_lex)	82.58	77.87	80.16	85.61
All-(head_POS)	82.66	77.88	80.20	85.89
All-(head_phr)	83.52	76.82	80.03	85.81
All-(cont_lex)	82.57	77.87	80.15	85.64
All-(cont_POS)	82.65	77.92	80.22	**86.09**
All-(gov)	82.69	78.34	**80.46**	85.91
All-(path)	78.39	67.96	72.80	85.69
All-(pos)	82.70	77.74	80.14	85.85
All-(pos+clau)	82.94	78.34	**80.57**	**86.19**
All-(pos+VP)	82.69	77.87	80.20	85.87
All-(pos+NP)	82.78	77.69	80.15	85.77
All-(pos+SBAR)	82.51	78.00	80.19	85.83
All-(pos+CC)	82.84	78.10	80.40	85.70
All-(pos+comma)	82.78	77.69	80.15	85.70
All-(pos+colon)	82.67	77.96	80.25	85.72
All-(pos+quote)	82.63	77.98	80.24	85.66
All-(pl+hl)	82.62	77.71	80.09	84.98
All-(pl+cl)	82.72	77.79	80.18	85.24
All-(v+gov)	82.93	77.81	80.29	85.85

	Prec.	Recall	F$_{\beta=1}$	Accu.
Iden.	82.56	78.72	80.59	-
clas.	-	-	-	87.16
Iden.+Clas.	72.68	69.16	70.87	-

Table 3: Performance of various feature combinations (top) and performance of each phase (bottom).

3 Experiments

To test the proposed method, we have experimented with CoNLL-2005 datasets (Wall Street sections 02-21 as training set, Charniak' trees). The results have been evaluated by using the *srl-eval.pl* script provided by the shared task organizers. For building classifiers, we utilized the Zhang le's MaxEnt toolkit [2], and the L-BFGS parameter estimation algorithm with Gaussian Prior smoothing.

Table 2 shows the different ways of reducing the number of argument candidates. The 2nd and 3rd columns (*#can.*, *%can.*) indicate the number of argument candidates and the percentage of argument candidates that satisfy each restriction on the training set. The 4th and 5th columns (*#arg.*, *%arg.*) indicate the number of correct arguments and the percentage of correct arguments that satisfy each restriction on the training set. The last column (F$_{\beta=1}$) indicates the performance of the identification task on the development set by applying each restriction.

In *no restriction*, All$_1$ extracts candidates from all the nonterminals's child nodes of a tree. All$_2$ filter the nonterminals which include at least one nonterminal child [3]. All$_3$ filter the nonterminals which include at least one nonterminal child and have distance from P$_p$. We use All$_3$ as a baseline.

In *restriction on clause boundary*, for example, *2/0* means that the left search boundary for identifying the argument is set to the left boundary of the second upper clause, and the right search boundary is set to the right boundary of the immediate clause.

In *restriction on tree distance*, for example, *7/1* means that it is possible to move up to 7 times upward ($d \leq 7$) through the parse tree from P$_p$ to P$_c$, and it is possible to move up to once downward ($d \leq 1$) through the parse tree from P$_p$ to P$_c$.

In *clause boundary & tree distance*, for example, *3/1,7/1* means the case when we use both the *clause boundary* (*3/1*) and the *tree distance* (*7/1*).

[2] http://www.nlplab.cn/zhangle/maxent_toolkit.html

[3] We ignore the nonterminals that have only pre-terminal children (e.g. in Figure 1, NP$_1$, CONJP, NP$_2$).

	Precision	Recall	$F_{\beta=1}$
Development	72.68%	69.16%	70.87
Test WSJ	74.69%	70.78%	72.68
Test Brown	64.58%	60.31%	62.38
Test WSJ+Brown	73.35%	69.37%	71.31

Test WSJ	Precision	Recall	$F_{\beta=1}$
Overall	74.69%	70.78%	72.68
A0	85.02%	81.53%	83.24
A1	73.98%	72.25%	73.11
A2	63.20%	57.57%	60.25
A3	62.96%	49.13%	55.19
A4	73.40%	67.65%	70.41
A5	100.00%	40.00%	57.14
AM-ADV	56.73%	50.00%	53.15
AM-CAU	70.21%	45.21%	55.00
AM-DIR	46.48%	38.82%	42.31
AM-DIS	70.95%	65.62%	68.18
AM-EXT	87.50%	43.75%	58.33
AM-LOC	44.09%	46.28%	45.16
AM-MNR	55.56%	52.33%	53.89
AM-MOD	97.59%	95.64%	96.61
AM-NEG	96.05%	95.22%	95.63
AM-PNC	40.68%	41.74%	41.20
AM-PRD	50.00%	20.00%	28.57
AM-REC	0.00%	0.00%	0.00
AM-TMP	70.11%	61.73%	65.66
R-A0	84.68%	83.93%	84.30
R-A1	73.33%	70.51%	71.90
R-A2	50.00%	31.25%	38.46
R-A3	0.00%	0.00%	0.00
R-A4	0.00%	0.00%	0.00
R-AM-ADV	0.00%	0.00%	0.00
R-AM-CAU	100.00%	25.00%	40.00
R-AM-EXT	0.00%	0.00%	0.00
R-AM-LOC	85.71%	57.14%	68.57
R-AM-MNR	16.67%	16.67%	16.67
R-AM-TMP	72.50%	55.77%	63.04
V	97.32%	97.32%	97.32

Table 4: Overall results (top) and detailed results on the WSJ test (bottom).

	Precision	Recall	$F_{\beta=1}$
one-phase	71.94	68.70	70.29
two-phase	72.68	69.16	70.87

Table 5: Performance of one-phase vs. two-phase.

According to the experimental results, we use *7/1 tree distance* restriction for all following experiments. By applying the restriction, we can remove about 47.3% (%can.=52.70%) of total argument candidates as compared with All_3. 93.90% (%arg.) corresponds to the upper bound on recall.

In order to estimate the relative contribution of each feature, we measure performance of each phase on the development set by leaving out one feature at a time, as shown in the top of Table 3. *Precision*, *Recall*, and $F_{\beta=1}$ represent the performance of the identification task, and *Accuracy* represent the performance of the classification task only with 100% correct argument identification respectively. *All* represents the performance of the experiment when all 26 features introduced by section 2.2 are considered. Finally, for identification, we use 24 features except *gov* and *pos+clau*, and obtain an $F_{\beta=1}$ of 80.59%, as shown in the bottom of Table 3. Also, for classification, we use 23 features except *pred_type*, *cont_POS*, and *pos+clau*, and obtain an *Accuracy* of 87.16%.

Table 4 presents our best system performance on the development set, and the performance of the same system on the test set. Table 5 shows the performance on the development set using the one-phase method and the two-phase method respectively. The one-phase method is implemented by incorporating the identification into the classification. *one-phase* shows the performance of the experiment when 25 features except *pos+clau* are used. Experimental results show that the two-phase method is better than the one-phase method in our evaluation.

4 Conclusion

We have presented a two-phase SRL method based on a ME model. In the two-phase method, in order to improve the performance of identification that dominate the overall performance, we have performed pre-processing. Experimental results show that our system obtains an $F_{\beta=1}$ of 72.68% on the WSJ test and that the introduction of pre-processing improves the performance, as compared with the case when all parse constituents are considered.

References

Xavier Carreras and Lluís Màrquez. 2005. *Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling*. Proceedings of CoNLL-2005.

Daniel Gildea and Daniel Jurafsky. 2002. *Automatic Labeling of Semantic Roles*. Computational Linguistics.

Joon-Ho Lim, Young-Sook Hwang, So-Young Park and Hae-Chang Rim. 2004. *Semantic Role Labeling using Maximum Entropy Model*. Proceedings of CoNLL.

Sameer Pradhan, Kadri Hacioglu, Valerie Krugler, Wayne Ward, James H. Martin and Daniel Jurafsky. 2003. *Shallow Semantic Parsing Using Support Vector Machines*. Technical Report, TR-CSLR-2003-03.