Multilingual Dependency-based Syntactic and Semantic Parsing

Wanxiang Che, Zhenghua Li, Yongqiang Li, Yuhang Guo, Bing Qin, Ting Liu
Information Retrieval Lab
School of Computer Science and Technology
Harbin Institute of Technology, China, 150001
{car, lzh, yqli, yhguo, qinb, tliu}@ir.hit.edu.cn

Abstract

Our CoNLL 2009 Shared Task system includes three cascaded components: syntactic parsing, predicate classification, and semantic role labeling. A pseudo-projective high-order graph-based model is used in our syntactic dependency parser. A support vector machine (SVM) model is used to classify predicate senses. Semantic role labeling is achieved using maximum entropy (MaxEnt) model based semantic role classification and integer linear programming (ILP) based post inference. Finally, we win the first place in the joint task, including both the closed and open challenges.

1 System Architecture

Our CoNLL 2009 Shared Task (Hajič et al., 2009): multilingual syntactic and semantic dependencies system includes three cascaded components: syntactic parsing, predicate classification, and semantic role labeling.

2 Syntactic Dependency Parsing

We extend our CoNLL 2008 graph-based model (Che et al., 2008) in four ways:

1. We use bigram features to choose multiple possible syntactic labels for one arc, and decide the optimal label during decoding.

2. We extend the model with sibling features (McDonald, 2006).

3. We extend the model with grandchildren features. Rather than only using the left-most and right-most grandchildren as Carreras (2007) and Johansson and Nugues (2008) did, we use all left and right grandchildren in our model.

4. We adopt the pseudo-projective approach introduced in (Nivre and Nilsson, 2005) to handle the non-projective languages including Czech, German and English.

2.1 Syntactic Label Determining

The model of (Che et al., 2008) decided one label for each arc before decoding according to unigram features, which caused lower labeled attachment score (LAS). On the other hand, keeping all possible labels for each arc made the decoding inefficient. Therefore, in the system of this year, we adopt approximate techniques to compromise, as shown in the following formulas.

$$\mathbf{f}^{lbl}_{uni}(h,c,l) = \mathbf{f}^{lbl}_{1}(h,1,d,l) \cup \mathbf{f}^{lbl}_{1}(c,0,d,l)$$

$$L_1(h,c) = \arg\max^{K_1}_{l \in L}(\mathbf{w} \cdot \mathbf{f}^{lbl}_{uni}(h,c,l))$$

$$\mathbf{f}^{lbl}_{bi}(h,c,l) = \mathbf{f}^{lbl}_{2}(h,c,l)$$

$$L_2(h,c) = \arg\max^{K_2}_{l \in L_1(h,c)}(\mathbf{w} \cdot \{\mathbf{f}^{lbl}_{uni} \cup \mathbf{f}^{lbl}_{bi}\})$$

For each arc, we firstly use unigram features to choose the K_1-best labels. The second parameter of $\mathbf{f}^{lbl}_1(\cdot)$ indicates whether the node is the head of the arc, and the third parameter indicates the direction. L denotes the whole label set. Then we re-rank the labels by combining the bigram features, and choose K_2-best labels. During decoding, we only use the K_2 labels chosen for each arc ($K_2 \ll K_1 < |L|$).

2.2 High-order Model and Algorithm

Following the Eisner (2000) algorithm, we use spans as the basic unit. A span is defined as a substring of the input sentence whose sub-tree is already produced. Only the start or end words of a span can link with other spans. In this way, the algorithm parses the left and the right dependence of a word independently, and combines them in the later stage.

We follow McDonald (2006)'s implementation of first-order Eisner parsing algorithm by modifying its scoring method to incorporate high-order features. Our extended algorithm is shown in Algorithm 1.

There are four different span-combining operations. Here we explain two of them that correspond to right-arc ($s < t$), as shown in Figure 1 and 2. We

Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL): Shared Task, pages 49–54,
Boulder, Colorado, June 2009.

Algorithm 1 High-order Eisner Parsing Algorithm

1: $C[s][s][c] = 0, 0 \leq s \leq N, c \in cp, icp$ # cp: complete; icp: incomplete
2: **for** $j = 1$ to N **do**
3: **for** $s = 0$ to N **do**
4: $t = s + j$L
5: **if** $t > N$ **then**
6: break
7: **end if**
 # Create incomplete spans
8: $C[s][t][icp] = \max_{s \leq r < t; l \in L_2(s,t)}(C[s][r][cp] + C[t][r+1][cp] + S_{icp}(s, r, t, l))$
9: $C[t][s][icp] = \max_{s \leq r < t; l \in L_2(t,s)}(C[s][r][cp] + C[t][r+1][cp] + S_{icp}(t, r, s, l))$
 # Create complete spans
10: $C[s][t][cp] = \max_{s < r \leq t; l = C[s][r][icp].label}(C[s][r][icp] + C[r][t][cp] + S_{cp}(s, r, t, l))$
11: $C[t][s][cp] = \max_{s \leq r < t; l = C[t][r][icp].label}(C[r][s][cp] + C[t][r][icp] + S_{cp}(t, r, s, l))$
12: **end for**
13: **end for**

follow the way of (McDonald, 2006) and (Carreras, 2007) to represent spans. The other two operations corresponding to left-arc are similar.

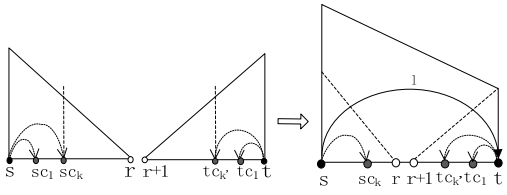

Figure 1: Combining two spans into an incomplete span

Figure 1 illustrates line 8 of the algorithm in Algorithm 1, which combines two complete spans into an incomplete span. A complete span means that only the head word can link with other words further, noted as "$\rightarrow$" or "$\leftarrow$". An incomplete span indicates that both the start and end words of the span will link with other spans in the future, noted as "$\dashrightarrow$" or "$\dashleftarrow$". In this operation, we combine two smaller spans, $sp_{s \rightarrow r}$ and $sp_{r+1 \leftarrow t}$, into $sp_{s \dashrightarrow t}$ with adding $arc_{s \rightarrow t}$. As shown in the following formulas, the score of $sp_{s \dashrightarrow t}$ is composed of three parts: the score of $sp_{s \rightarrow r}$, the score of $sp_{r+1 \leftarrow t}$, and the score of adding $arc_{s \rightarrow t}$. The score of $arc_{s \rightarrow t}$ is determined by four different feature sets: unigram features, bigram features, sibling features and left grandchildren features (or inside grandchildren features, meaning that the grandchildren lie between s and t). Note that the sibling features are only related to the nearest sibling node of t, which is denoted as sc_k here. And the inside grandchildren features are related to all the children of t. This is different from the models used by Carreras (2007) and Johansson and Nugues (2008). They only used the left-most child of t, which is $tc_{k'}$ here.

$$\mathbf{f}_{icp}(s, r, t, l) = \mathbf{f}_{uni}(s, t, l) \cup \mathbf{f}_{bi}(s, t, l) \cup \mathbf{f}_{sib}(s, sc_k, t) \cup \{\bigcup_{i=1}^{k'} \mathbf{f}_{grand}(s, t, tc_i, l)\}$$

$$S_{icp}(s, r, t, l) = \mathbf{w} \cdot \mathbf{f}_{icp}(s, r, t, l)$$

$$S(sp_{s \dashrightarrow t}) = S(sp_{s \rightarrow r}) + S(sp_{r+1 \leftarrow t}) + S_{icp}(s, r, t, l)$$

In Figure 2 we combine $sp_{s \dashrightarrow r}$ and $sp_{r \rightarrow t}$ into $sp_{s \rightarrow t}$, which explains line 10 in Algorithm 1. The score of $sp_{s \rightarrow t}$ also includes three parts, as shown in the following formulas. Although there is no new arc added in this operation, the third part is necessary because it reflects the right (or called outside) grandchildren information of $arc_{s \rightarrow r}$.

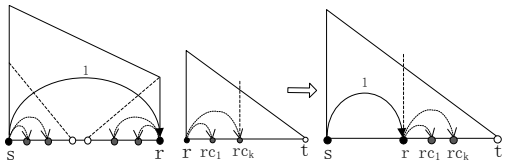

Figure 2: Combining two spans into a complete span

$$\mathbf{f}_{cp}(s, r, t, l) = \bigcup_{i=1}^{k} \mathbf{f}_{grand}(s, r, rc_i, l)$$

$$S_{cp}(s, r, t, l) = \mathbf{w} \cdot \mathbf{f}_{cp}(s, r, t, l)$$

$$S(sp_{s \rightarrow t}) = S(sp_{s \dashrightarrow r}) + S(sp_{r \rightarrow t}) + S_{cp}(s, r, t, l)$$

2.3 Features

As shown above, features used in our model can be decomposed into four parts: unigram features, bigram features, sibling features, and grandchildren features. Each part can be seen as two different sets: arc-related and label-related features, except sibling features, because we do not consider labels when using sibling features. Arc-related features can be understood as back-off of label-related features. Actually, label-related features are gained by simply attaching the label to the arc-features.

The unigram and bigram features used in our model are similar to those of (Che et al., 2008), except that we use bigram label-related features. The sibling features we use are similar to those of (McDonald, 2006), and the grandchildren features are similar to those of (Carreras, 2007).

3 Predicate Classification

The predicate classification is regarded as a supervised word sense disambiguation (WSD) task here. The task is divided into four steps:

1. Target words selection: predicates with multiple senses appearing in the training data are selected as target words.

2. Feature extraction: features in the context around these target words are extracted as shown in Table 4. The detailed explanation about these features can be found from (Che et al., 2008).

3. Classification: for each target word, a Support Vector Machine (SVM) classifier is used to classify its sense. As reported by Lee and Ng (2002) and Guo et al. (2007), SVM shows good performance on the WSD task. Here libsvm (Chang and Lin, 2001) is used. The linear kernel function is used and the trade off parameter C is 1.

4. Post processing: for each predicate in the test data which does not appear in the training data, its first sense in the frame files is used.

4 Semantic Role Labeling

The semantic role labeling (SRL) can be divided into two separate stages: semantic role classification (SRC) and post inference (PI).

During the SRC stage, a Maximum entropy (Berger et al., 1996) classifier is used to predict the probabilities of a word in the sentence to be each semantic role. We add a virtual role "NULL" (presenting none of roles is assigned) to the roles set, so we do not need semantic role identification stage anymore. For a predicate of each language, two classifiers (one for noun predicates, and the other for verb predicates) predict probabilities of each word in a sentence to be each semantic role (including virtual role "NULL"). The features used in this stage are listed in Table 4.

Language	No-duplicated-roles
Catalan	arg0-agt, arg0-cau, arg1-pat, arg2-atr, arg2-loc
Chinese	A0, A1, A2, A3, A4, A5,
Czech	ACT, ADDR, CRIT, LOC, PAT, DIR3, COND
English	A0, A1, A2, A3, A4, A5,
German	A0, A1, A2, A3, A4, A5,
Japanese	DE, GA, TMP, WO
Spanish	arg0-agt, arg0-cau, arg1-pat, arg1-tem, arg2-atr, arg2-loc, arg2-null, arg4-des, argL-null, argM-cau, argM-ext, argM-fin

Table 1: No-duplicated-roles for different languages

The probability of each word to be a semantic role for a predicate is given by the SRC stage. The results generated by selecting the roles with the largest probabilities, however, do not satisfy some constrains. As we did in the last year's system (Che et al., 2008), we use the ILP (Integer Linear Programming) (Punyakanok et al., 2004) to get the global optimization, which is satisfied with three constrains:

C1: Each word should be labeled with one and only one label (including the virtual label "NULL").

C2: Roles with a small probability should never be labeled (except for the virtual role "NULL"). The threshold we use in our system is 0.3.

C3: Statistics show that some roles (except for the virtual role "NULL") usually appear once for a predicate. We impose a no-duplicate-roles constraint with a no-duplicate-roles list, which is constructed according to the times of semantic roles' duplication for each single predicate. Table 1 shows the no-duplicate-roles for different languages.

Our maximum entropy classifier is implemented with Maximum Entropy Modeling Toolkit[1]. The classifier parameters are tuned with the development data for different languages respectively. lp_solve 5.5[2] is chosen as our ILP problem solver.

[1] http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html

[2] http://sourceforge.net/projects/lpsolve

5 Experiments

5.1 Experimental Setup

We participate in the CoNLL 2009 shared task with all 7 languages: Catalan (Taulé et al., 2008), Chinese (Palmer and Xue, 2009), Czech (Hajič et al., 2006), English (Surdeanu et al., 2008), German (Burchardt et al., 2006), Japanese (Kawahara et al., 2002), and Spanish (Taulé et al., 2008). Besides the closed challenge, we also submitted the open challenge results. Our open challenge strategy is very simple. We add the SRL development data of each language into their training data. The purpose is to examine the effect of the additional data, especially for out-of-domain (ood) data.

Three machines (with 2.5GHz Xeon CPU and 16G memory) were used to train our models. During the peak time, Amazon's EC2 (Elastic Compute Cloud)[3] was used, too. Our system requires 15G memory at most and the longest training time is about 36 hours.

During training the predicate classification (PC) and the semantic role labeling (SRL) models, golden syntactic dependency parsing results are used. Previous experiments show that the PC and SRL test results based on golden parse trees are slightly worse than that based on cross trained parse trees. It is, however, a pity that we have no enough time and machines to do cross training for so many languages.

5.2 Results and Discussion

In order to examine the performance of the ILP based post inference (PI) for different languages, we adopt a simple PI strategy as baseline, which selects the most likely label (including the virtual label "NULL") except for those duplicate non-virtual labels with lower probabilities (lower than 0.5). Table 2 shows their performance on development data.

We can see that the ILP based post inference can improve the precision but decrease the recall. Except for Czech, almost all languages are improved. Among them, English benefits most.

The final system results are shown in Table 3. Comparing with our CoNLL 2008 (Che et al., 2008) syntactic parsing results on English[4], we can see that our new high-order model improves about 1%.

[3] http://aws.amazon.com/ec2/

[4] devel: 85.94%, test: 87.51% and ood: 80.73%

	Precision	Recall	F1
Catalan simple	78.68	77.14	77.90
Catalan ILP	79.42	76.49	77.93
Chinese simple	80.74	74.36	77.42
Chinese ILP	81.97	73.92	77.74
Czech simple	88.54	84.68	86.57
Czech ILP	89.23	84.05	86.56
English simple	83.03	83.55	83.29
English ILP	85.63	83.03	84.31
German simple	78.88	75.87	77.34
German ILP	82.04	74.10	77.87
Japanese simple	88.04	70.68	78.41
Japanese ILP	89.23	70.16	78.56
Spanish simple	76.73	75.92	76.33
Spanish ILP	77.71	75.34	76.51

Table 2: Comparison between different PI strategies

For the open challenge, because we did not modify the syntactic training data, its results are the same as the closed ones. We can, therefore, examine the effect of the additional training data on SRL. We can see that along with the development data are added into the training data, the performance on the in-domain test data is increased. However, it is interesting that the additional data is harmful to the ood test.

6 Conclusion and Future Work

Our CoNLL 2009 Shared Task system is composed of three cascaded components. The pseudo-projective high-order syntactic dependency model outperforms our CoNLL 2008 model (in English). The additional in-domain (devel) SRL data can help the in-domain test. However, it is harmful to the ood test. Our final system achieves promising results. In the future, we will study how to solve the domain adaptive problem and how to do joint learning between syntactic and semantic parsing.

Acknowledgments

This work was supported by National Natural Science Foundation of China (NSFC) via grant 60803093, 60675034, and the "863" National High-Tech Research and Development of China via grant 2008AA01Z144.

		Syntactic Accuracy (LAS)			Semantic Labeled F1			Macro F1 Score		
		devel	test	ood	devel	test	ood	devel	test	ood
Catalan	closed	86.65	86.56	—	77.93	77.10	—	82.30	81.84	—
	open	—			—	77.36		—	81.97	
Chinese	closed	75.73	75.49	—	77.74	77.15	—	76.79	76.38	—
	open	—			—	77.23		—	76.42	
Czech	closed	80.07	80.01	76.03	86.56	86.51	85.26	83.33	83.27	80.66
	open	—			—	86.57	85.21	—	83.31	80.63
English	closed	87.09	88.48	81.57	84.30	85.51	73.82	85.70	87.00	77.71
	open	—			—	85.61	73.66	—	87.05	77.63
German	closed	85.69	86.19	76.11	77.87	78.61	70.07	81.83	82.44	73.19
	open	—			—	78.61	70.09	—	82.44	73.20
Japanese	closed	92.55	92.57	—	78.56	78.26	—	85.86	85.65	—
	open	—			—	78.35		—	85.70	
Spanish	closed	87.22	87.33	—	76.51	76.47	—	81.87	81.90	—
	open	—			—	76.66		—	82.00	
Average	closed	—	85.23	77.90	—	79.94	76.38	—	82.64	77.19
	open					80.06	76.32		82.70	77.15

Table 3: Final system results

References

Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22.

Aljoscha Burchardt, Katrin Erk, Anette Frank, Andrea Kowalski, Sebastian Padó, and Manfred Pinkal. 2006. The SALSA corpus: a German corpus resource for lexical semantics. In *LREC-2006*.

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

Chih-Chung Chang and Chih-Jen Lin, 2001. *LIBSVM: a library for support vector machines*.

Wanxiang Che, Zhenghua Li, Yuxuan Hu, Yongqiang Li, Bing Qin, Ting Liu, and Sheng Li. 2008. A cascaded syntactic and semantic dependency parsing system. In *CoNLL-2008*.

Jason Eisner. 2000. Bilexical grammars and their cubic-time parsing algorithms. In *Advances in Probabilistic and Other Parsing Technologies*.

Yuhang Guo, Wanxiang Che, Yuxuan Hu, Wei Zhang, and Ting Liu. 2007. HIT-IR-WSD: A wsd system for english lexical sample task. In *SemEval-2007*.

Jan Hajič, Jarmila Panevová, Eva Hajičová, Petr Sgall, Petr Pajas, Jan Štěpánek, Jiří Havelka, Marie Mikulová, and Zdeněk Žabokrtský. 2006. Prague Dependency Treebank 2.0.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *CoNLL-2009*.

Richard Johansson and Pierre Nugues. 2008. Dependency-based semantic role labeling of PropBank. In *EMNLP-2008*.

Daisuke Kawahara, Sadao Kurohashi, and Kôiti Hasida. 2002. Construction of a Japanese relevance-tagged corpus. In *LREC-2002*.

Yoong Keok Lee and Hwee Tou Ng. 2002. An empirical evaluation of knowledge sources and learning algorithms for word sense disambiguation. In *EMNLP-2002*.

Ryan McDonald. 2006. *Discriminative Learning and Spanning Tree Algorithms for Dependency Parsing*. Ph.D. thesis, University of Pennsylvania.

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

Martha Palmer and Nianwen Xue. 2009. Adding semantic roles to the Chinese Treebank. *Natural Language Engineering*, 15(1).

Vasin Punyakanok, Dan Roth, Wen-tau Yih, and Dav Zimak. 2004. Semantic role labeling via integer linear programming inference. In *Coling-2004*.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *CoNLL-2008*.

Mariona Taulé, Maria Antònia Martí, and Marta Recasens. 2008. AnCora: Multilevel Annotated Corpora for Catalan and Spanish. In *LREC-2008*.

	Catalan	Chinese	Czech	English	German	Japanese	Spanish
ChildrenPOS			♦ ◇				♦◇
ChildrenPOSNoDup				♦ ◇			♦ ◇
ConstituentPOSPattern	♦ ◇	♦ ◇	♦ ◇	♦ ◇	♦ ◇	♦ ◇	♦ ◇
ConstituentPOSPattern+DepRelation	♦ ◇		♦ ◇		♦ ◇		
ConstituentPOSPattern+DepwordLemma	♦ ◇		♦ ◇		♦ ◇		
ConstituentPOSPattern+HeadwordLemma		♦ ◇	♦ ◇	♦ ◇		♦ ◇	♦ ◇
DepRelation	▲ △ ♦ ◇	▲ △ ♦ ◇	▲ △ ♦ ◇	▲ △ ♦ ◇	▲ △	♦ ◇	▲ △ ♦ ◇
DepRelation+DepwordLemma	♦ ◇		♦ ◇				
DepRelation+Headword	▲ △	▲ △	▲	▲ △	▲ △		▲
DepRelation+HeadwordLemma	♦ ◇		♦ ◇		♦ ◇		♦ ◇
DepRelation+HeadwordLemma+DepwordLemma		♦ ◇	♦ ◇	♦ ◇	♦ ◇	♦ ◇	♦ ◇
DepRelation+HeadwordPOS	▲ △	▲ △	▲ △	▲ △	▲ △		▲
Depword	♦ ◇						♦ ◇
DepwordLemma	♦ ◇	♦ ◇	♦ ◇	♦ ◇		♦ ◇	♦ ◇
DepwordLemma+HeadwordLemma	♦ ◇				♦ ◇	♦ ◇	
DepwordLemma+RelationPath		♦ ◇	♦ ◇	♦ ◇		♦ ◇	♦ ◇
DepwordPOS	▲ △	▲ △	▲ △ ♦ ◇	▲ △	▲ △ ♦ ◇		▲ △
DepwordPOS+HeadwordPOS	♦ ◇		♦ ◇				
DownPathLength			♦ ◇				♦ ◇
FirstLemma	♦ ◇	♦ ◇	♦ ◇	♦ ◇		♦ ◇	♦ ◇
FirstPOS			♦ ◇		♦ ◇		
FirstPOS+DepwordPOS	♦ ◇		♦ ◇		♦ ◇		
FirstWord	♦ ◇				♦ ◇		
Headword	▲ △	▲ △	▲ △	▲ △	▲ △ ♦ ◇		▲
HeadwordLemma	▲ △ ♦ ◇	▲ △ ♦ ◇	▲ △ ♦ ◇	▲ △ ♦ ◇	▲ △ ♦ ◇	♦ ◇	▲
HeadwordLemma+RelationPath		♦ ◇	♦ ◇	♦ ◇	♦ ◇	♦ ◇	♦ ◇
HeadwordPOS	▲ △	▲ △	▲ △ ♦ ◇	▲ △ ♦ ◇	▲ △ ♦ ◇		▲ △
LastLemma	♦ ◇	♦ ◇	♦ ◇	♦ ◇		♦ ◇	
LastPOS	♦ ◇		♦ ◇				
LastWord	♦ ◇						
Path	♦ ◇	♦ ◇	♦ ◇	♦ ◇	♦ ◇	♦ ◇	
Path+RelationPath		♦ ◇	♦ ◇	♦ ◇	♦ ◇	♦ ◇	
PathLength		♦ ◇	♦ ◇	♦ ◇	♦ ◇	♦ ◇	♦ ◇
PFEAT	▲ △		▲ △				▲ △
PFEATSplit	▲ △ ♦ ◇		▲ △ ♦ ◇		▲ △ ♦ ◇		▲ △ ♦ ◇
PFEATSplitRemoveNULL	▲ △				▲ △		▲ △
PositionWithPredicate	♦ ◇	♦ ◇	♦ ◇	♦ ◇		♦ ◇	
Predicate	▲ △ ♦ ◇	▲ △	▲ △ ♦ ◇	▲ △	▲ △		▲ △ ♦ ◇
Predicate+PredicateFamilyship		♦ ◇	♦ ◇	♦ ◇		♦ ◇	♦ ◇
PredicateBagOfPOSNumbered	△	▲ △			▲ △		▲ △
PredicateBagOfPOSNumberedWindow5	▲ △	▲ △	▲	△	▲ △		▲ △
PredicateBagOfPOSOrdered	▲ △	▲ △	▲ △		▲ △		▲
PredicateBagOfPOSOrderedWindow5	▲ △	▲ △	▲ △	▲ △	▲ △		▲ △
PredicateBagOfPOSWindow5	▲	▲ △	▲ △	▲ △	▲ △		▲
PredicateBagOfWords		△	▲ △	▲	▲ △		▲ △
PredicateBagOfWordsAndIsDesOfPRED	▲ △	▲	△	△	▲ △		▲ △
PredicateBagOfWordsOrdered	△	▲ △	▲ △	△	▲ △		▲ △
PredicateChildrenPOS	▲ △ ♦ ◇	▲ △	▲ △	▲ △	▲ △		▲ △ ♦ ◇
PredicateChildrenPOSNoDup	▲ △	▲ △	▲ △	▲ △	▲ △		▲ △
PredicateChildrenREL	▲ △ ♦ ◇	▲ △	▲ △	▲ △	▲ △ ♦ ◇		▲ △
PredicateChildrenRELNoDup	▲ △ ♦ ◇	▲ △	▲ △	▲ △	▲ △ ♦ ◇		▲ △
PredicateFamilyship					♦ ◇		
PredicateLemma	▲ △ ♦ ◇	▲ △ ♦ ◇	▲ △ ♦ ◇	▲ △ ♦ ◇	▲ △ ♦ ◇	♦ ◇	▲ △ ♦ ◇
PredicateLemma+PredicateFamilyship	♦ ◇		♦ ◇		♦ ◇		
PredicateSense	♦ ◇	♦ ◇	♦ ◇	♦ ◇	♦ ◇	♦ ◇	♦ ◇
PredicateSense+DepRelation			♦ ◇		♦ ◇		
PredicateSense+DepwordLemma			♦ ◇		♦ ◇		
PredicateSense+DepwordPOS			♦ ◇		♦ ◇		
PredicateSiblingsPOS	▲ △	▲ △	▲	▲ △	▲ △		▲ △
PredicateSiblingsPOSNoDup	▲ △ ♦ ◇	▲ △	▲ △	▲ △	▲ △		▲ △ ♦ ◇
PredicateSiblingsREL	▲ △ ♦ ◇	▲ △	▲ △	▲ △	▲ △		▲ △
PredicateSiblingsRELNoDup	▲ △	▲ △ ♦ ◇	△	▲ △	▲ △ ♦ ◇		▲ △ ♦ ◇
PredicateVoiceEn				▲ △			
PredicateWindow5Bigram		▲ △	▲ △	▲ △			▲ △
PredicateWindow5BigramPOS	▲ △	▲ △	▲ △	▲ △	▲ △		▲ △
RelationPath	♦ ◇	♦ ◇	♦ ◇	♦ ◇	♦ ◇	♦ ◇	♦ ◇
SiblingsPOS					♦ ◇		♦ ◇
SiblingsREL	♦						
SiblingsRELNoDup			♦ ◇		♦ ◇		
UpPath		♦ ◇	♦ ◇	♦ ◇		♦	
UpPathLength			♦ ◇				
UpRelationPath	♦ ◇				♦ ◇		♦ ◇
UpRelationPath+HeadwordLemma		♦ ◇	♦ ◇	♦ ◇		♦ ◇	

Table 4: Features that are used in predicate classification (PC) and semantic role labeling (SRL). ▲: noun predicate PC, △: verb predicate PC, ♦: noun predicate SRL, ◇: verb predicate SRL