A Cascaded Syntactic and Semantic Dependency Parsing System

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

Abstract

We describe our CoNLL 2008 Shared Task system in this paper. The system includes two cascaded components: a syntactic and a semantic dependency parsers. A first-order projective MSTParser is used as our syntactic dependency parser. In order to overcome the shortcoming of the MST-Parser, that it cannot model more global information, we add a relabeling stage after the parsing to distinguish some confusable labels, such as ADV, TMP, and LOC. Besides adding a predicate identification and a classification stages, our semantic dependency parsing simplifies the traditional four stages semantic role labeling into two: a maximum entropy based argument classification and an ILP-based post inference. Finally, we gain the overall labeled macro F1 = 82.66, which ranked the second position in the closed challenge.

1 System Architecture

Our CoNLL 2008 Shared Task (Surdeanu et al., 2008) participating system includes two cascaded components: a syntactic and a semantic dependency parsers. They are described in Section 2 and 3 respectively. Their experimental results are shown in Section 4. Section 5 gives our conclusion and future work.

2 Syntactic Dependency Parsing

MSTParser (McDonald, 2006) is selected as our basic syntactic dependency parser. It views the syntactic dependency parsing as a problem of finding maximum spanning trees (MST) in directed graphs. MSTParser provides the state-of-the-art performance for both projective and non-projective tree banks.

2.1 Features

The score of each labeled arc is computed through the Eq. (1) in MSTParser.

$$score(h, c, l) = \mathbf{w} \cdot \mathbf{f}(h, c, l) \tag{1}$$

where node h represents the head node of the arc, while node c is the dependent node (or child node). l denotes the label of the arc.

There are three major differences between our feature set and McDonald (2006)'s:

1) We use the lemma as a generalization feature of a word, while McDonald (2006) use the word's prefix.

2) We add two new features: "bet-pos-h-same-num" and "bet-pos-c-same-num". They represent the number of nodes which locate between node h and node c and whose POS tags are the same with h and c respectively.

3) We use more back-off features than McDonald (2006) by completely enumerating all of the possible combinatorial features.

2.2 Relabeling

By observing the current results of MSTParser on the development data, we find that the performance of some labels are far below average, such as ADV, TMP, LOC. We think the main reason lies in that MSTParser only uses local features restricted to a single arc (as shown in Eq. (1)) and fails to use more global information. Consider two sentences: "I read books in the room." and "I read books in the afternoon.". It is hard to correctly label the arc

Deprel	Total	Mislabeled as
NMOD	8,922	NAME [0.4], DEP [0.4], LOC [0.1], AMOD [0.1]
OBJ	1,728	TMP [0.5], ADV [0.4], OPRD[0.3]
ADV	1,256	TMP [2.9], LOC [2.3], MNR [1.8], DIR [1.5]
NAME	1,138	NMOD [2.2]
VC	953	PRD [0.9]
DEP	772	NMOD [4.0]
TMP	755	ADV [9.9], LOC [6.5]
LOC	556	ADV [12.6], NMOD [7.9], TMP [5.9]
AMOD	536	ADV [2.2]
PRD	509	VC [4.7]
APPO	444	NMOD [2.5]
OPRD	373	OBJ [4.6]
DIR	119	ADV [18.5]
MNR	109	ADV [28.4]

Table 1: Error Analysis of Each Label

between "read" and "in" unless we know the object of "in".

We count the errors of each label, and show the top ones in Table 1. "Total" refers to the total number of the corresponding label in the development data. The column of "Mislabeled as" lists the labels that an arc may be mislabeled as. The number in brackets shows the percentage of mislabeling. As shown in the table, some labels are often confusable with each other, such as ADV, LOC and TMP.

2.3 Relabeling using Maximum Entropy Classifier

We constructed two confusable label set which have a higher mutual mislabeling proportion: (NMOD, LOC, ADV, TMP, MNR, DIR) and (OBJ, OPRD). A maximum entropy classifier is used to relabel them.

Features are shown in Table 2. The first column lists local features, which contains information of the head node h and the dependent node c of an arc. "+ dir dist" means that conjoining existing features with arc direction and distance composes new features. The second column lists features using the information of node c's children. "word_c_c" represents form or lemma of one child of the node c. "dir_c" and "dist_c" represents the direction and distance of the arc which links node c to its child. The back-off technique is also used on these features.

Local features (+ dir dist)	Global features (+ dir_c dist_c)
word_h word_c	word_h word_c word_c_c

Table 2: Relabeling Feature Set (+ dir dist)

3 Semantic Dependency Parsing

3.1 Architecture

The whole procedure is divided into four separate stages: Predicate Identification, Predicate Classification, Semantic Role Classification, and Post Inference.

During the Predicate Identification stage we examine each word in a sentence to discover target predicates, including both noun predicates (from NomBank) and verb predicates (from PropBank). In the Predicate Classification stage, each predicate is assigned a certain sense number. For each predicate, the probabilities of a word in the sentence to be each semantic role are predicted in the Semantic Role Classification stage. Maximum entropy model is selected as our classifiers in these stages. Finally an ILP (Integer Linear Programming) based method is adopted for post inference (Punyakanok et al., 2004).

3.2 Predicate Identification

The predicate identification is treated as a binary classification problem. Each word in a sentence is predicted to be a predicate or not to be. A set of features are extracted for each word, and an optimized subset of them are adopted in our final system. The following is a full list of the features:

DEPREL (a1): Type of relation to the parent.

WORD (a21), POS (a22), LEMMA (a23), HEAD (a31), HEAD_POS (a32), HEAD_LEMMA (a33): The forms, POS tags and lemmas of a word and it's headword (parent) .

FIRST_WORD (a41), FIRST_POS (a42), FIRST_LEMMA (a43), LAST_WORD (a51), LAST_POS (a52), LAST_LEMMA (a53): A corresponding "constituent" for a word consists of all descendants of it. The forms, POS tags and lemmas of both the first and the last words in the constituent are extracted.

POS_PAT (a6): A "POS pattern" is produced for the corresponding constituent as follows: a POS bag is produced with the POS tags of the words in the constituent except for the first and the last ones, duplicated tags removed and the original order ignored. Then we have the POS_PAT feature by combining the POS tag of the first word, the bag and the POS tag of the last word.

CHD_POS (a71), CHD_POS_NDUP (a72), CHD_REL (a73), CHD_REL_NDUP (a74): The POS tags of the child words are joined together to form feature CHD_POS. With adjacently duplicated tags reduced to one, feature CHD_POS_NDUP is produced. Similarly we can get CHD_REL and CHD_REL_NDUP too, with the relation types substituted for the POS tags.

SIB_REL (a81), SIB_REL_NDUP (a82), SIB_POS (a83), SIB_POS_NDUP (a84): Sibling words (including the target word itself) and the corresponding dependency relations (or POS tags) are considered as well. Four features are formed similarly to those of child words.

VERB_VOICE (a9): Verbs are examined for voices: if the headword lemma is either "be" or "get", or else the relation type is "APPO", then the verb is considered passive, otherwise active.

Also we used some "combined" features which are combinations of single features. The final optimized feature set is (a1, a21, a22, a31, a32, a41, a42, a51, a52, a6, a72, a73, a74, a81, a82, a83, a1+a21, a21+a31, a21+a6, a21+a74, a73+a81, a81+a83).

3.3 Predicate Classification

After predicate identification is done, the resulting predicates are processed for sense classification. A classifier is trained for each predicate that has multiple senses on the training data (There are totally 962 multi-sense predicates on the training corpus, taking up 14% of all) In additional to those features described in the predicate identification section, some new ones relating to the predicate word are introduced:

BAG_OF_WORD (b11), BAG_OF_WORD_O (b12): All words in a sentence joined, namely "Bag of Words". And an "ordered" version is introduced where each word is prefixed with a letter "L", "R" or "T" indicating it's to the left or right of the predicate or is the predicate itself.

BAG_OF_POS_O (b21), BAG_OF_POS_N (b22): The POS tags prefixed with "L", "R" or "T" indicating the word position joined together, namely "Bag of POS (Ordered)". With the prefixed letter changed to a number indicating the distance to the predicate (negative for being left to the predicate and positive for right), another feature is formed, namely "Bag of POS (Numbered)".

WIND5_BIGRAM (b3): 5 closest words from both left and right plus the predicate itself, in total 11 words form a "window", within which bigrams are enumerated.

The final optimized feature set for the task of predicate classification is (a1, a21, a23, a71, a72, a73, a74, a81, a82, a83, a84, a9, b11, b12, b22, b3, a71+a9).

3.4 Semantic Role Classification

In our system, the identification and classification of semantic roles are achieved in a single stage (Liu et al., 2005) through one single classifier (actually two, one for noun predicates, and the other for verb predicates). Each word in a sentence is given probabilities to be each semantic role (including none of the these roles) for a predicate. Features introduced in addition to those of the previous subsections are the following:

POS_PATH (c11), REL_PATH (c12): The "POS Path" feature consists of POS tags of the words along the path from a word to the predicate. Other than "Up" and "Down", the "Left" and "Right" direction of the path is added. Similarly, the "Relation Path" feature consists of the relation types along the same path.

UP_PATH (c21), UP_REL_PATH (c22): "Upstream paths" are parts of the above paths that stop at the common ancestor of a word and the predicate.

PATH_LEN (c3): Length of the paths

POSITION (c4): The relative position of a word to the predicate: Left or Right.

PRED_FAMILYSHIP (c5): "Familyship relation" between a word and the predicate, being one of "self", "child", "descendant", "parent", "ancestor", "sibling", and "not-relative".

PRED_SENSE (c6): The lemma plus sense number of the predicate

As for the task of semantic role classification, the features of the predicate word in addition to those of the word under consideration can also be used; we mark features of the predicate with an extra 'p'. For example, the head word of the current word is represented as a31, and the head word of the predicate is represented as pa31. So, with no doubt for the representation, our final optimized feature set for the task of semantic role classification is (a1, a23, a33, a43, a53, a6, c11, c12, c21, c3, c4, c6, pa23, pa71, pa73, pa83, a1+a23+a33, a21+c5, a23+c12, a33+c12, a33+c22, a6+a33, a73+c5, c11+c12, pa71+pa73).

3.5 ILP-based Post Inference

The final semantic role labeling result is generated through an ILP (Integer Linear Programming) based post inference method. An ILP problem is formulated with respect to the probability given by the above stage. The final labeling is formed at the same time when the problem is solved.

Let W be the set of words in the sentence, and R be the set of semantic role labels. A virtual label "NULL" is also added to R, representing "none of the roles is assigned".

For each word $w \in W$ and semantic role label $r \in R$ we create a binary variable $v_{wr} \in (0, 1)$, whose value indicates whether or not the word w is labeled as label r. p_{wr} denotes the possibility of word w to be labeled as role r. Obviously, when objective function $f = \sum_{w,r} \log(p_{wr} \cdot v_{wr})$ is maximized, we can read the optimal labeling for a predicate from the assignments to the variables v_{wr}. There are three constrains used in our system:

C1: Each relation should be and only be labeled with one label (including the virtual label "NULL"), i.e.:

$$\sum_r v_{wr} = 1$$

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, which is optimized from the development data. i.e.:

$$v_{wr} = 0, \text{ if } p_{wr} < 0.3 \text{ and } r \neq \text{“NULL”}$$

C3: Statistics shows that the most roles (except for the virtual role "NULL") usually appear only once for a predicate, except for some rare exception. So we impose a no-duplicated-roles constraint with an exception list, which is constructed according to the times of semantic roles' duplication for each single predicate (different senses of a predicate are considered different) and the ratio of duplication to non-duplication.

$$\begin{aligned} &\textstyle\sum_r v_{wr} \leq 1, \\ &\text{if } <p,r> \notin \{<p,r> | p \in P, r \in R; \\ &\qquad \tfrac{d_{pr}}{c_{pr} - d_{pr}} > 0.3 \wedge d_{pr} > 10\} \end{aligned} \tag{2}$$

where P is the set of predicates; c_{pr} denotes the count of words in the training corpus, which are labeled as $r \in R$ for predicate $p \in P$; while d_{pr} denotes something similar to c_{pr}, but what taken into account are only those words labeled with r, and there are more than one roles within the sentence for the same predicate. Table 3 lists the complete exception set, which has a size of only 10.

Predicate Type	Predicate	Label
Noun	president.01	A3
Verb	match.01	A1
Verb	tie.01	A1
Verb	link.01	A1
Verb	rate.01	A0
Verb	rate.01	A2
Verb	attach.01	A1
Verb	connect.01	A1
Verb	fit.01	A1
Noun	trader.01	SU

Table 3: No-duplicated-roles constraint exception list (obtained by Eq. (2))

4 Experiments

The original MSTParser[1] is implemented in Java. We were confronted with memory shortage when trying to train a model with the entire CoNLL 2008 training data with 4GB memory. Therefore, we rewrote it with C++ which can manage the memory more exactly. Since the time was limited, we only rewrote the projective part without considering second-order parsing technique.

Our maximum entropy classifier is implemented with Maximum Entropy Modeling Toolkit[2]. The classifier parameters: gaussian prior and iterations, are tuned with the development data for different stages respectively.

lp_solve 5.5[3] is chosen as our ILP problem solver during the post inference stage.

The training time of the syntactic and the semantic parsers are 22 and 5 hours respectively, on all training data, with 2.0GHz Xeon CPU and 4G memory. While the prediction can be done within 10 and 5 minutes on the development data.

4.1 Syntactic Dependency Parsing

The experiments on development data show that relabeling process is helpful, which improves the

[1]http://sourceforge.net/projects/mstparser

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

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

	Precision (%)	Recall (%)	F1
Pred Identification	91.61	91.36	91.48
Pred Classification	86.61	86.37	86.49

Table 4: The performance of predicate identification and classification

	Precision (%)	Recall (%)	F1
Simple	81.02	76.00	78.43
ILP-based	82.53	75.26	78.73

Table 5: Comparison between different post inference strategies

LAS performance from 85.41% to 85.94%. The final syntactic dependency parsing performances on the WSJ and the Brown test data are 87.51% and 80.73% respectively.

4.2 Semantic Dependency Parsing

The semantic dependency parsing component is based on the last syntactic dependency parsing component. All stages of the system are trained with the closed training corpus, while predicted against the output of the syntactic parsing.

Performance for predicate identification and classification is given in Table 4, wherein the classification is done on top of the identification.

Semantic role classification and the post inference are done on top of the result of predicate identification and classification. The final performance is presented in Table 5. A simple post inference strategy is given for comparison, where the most possible label (including the virtual label "NULL") is select except for those duplicated non-virtual labels with lower probabilities (lower than 0.5). Our ILP-based method produces a gain of 0.30 with respect to the F1 score.

The final semantic dependency parsing performance on the development and the test (WSJ and Brown) data are shown in Table 6.

	Precision (%)	Recall (%)	F1
Development	82.53	75.26	78.73
Test (WSJ)	82.67	77.50	80.00
Test (Brown)	64.38	68.50	66.37

Table 6: Semantic dependency parsing performances

4.3 Overall Performance

The overall macro scores of our syntactic and semantic dependency parsing system are 82.38%, 83.78% and 73.57% on the development and two test (WSJ and Brown) data respectively, which is ranked the second position in the closed challenge.

5 Conclusion and Future Work

We present our CoNLL 2008 Shared Task system which is composed of two cascaded components: a syntactic and a semantic dependency parsers, which are built with some state-of-the-art methods. Through a fine tuning features and parameters, the final system achieves promising results. In order to improve the performance further, we will study how to make use of more resources and tools (open challenge) and how to do joint learning between syntactic and semantic parsing.

Acknowledgments

The authors would like to thank the reviewers for their helpful comments. This work was supported by National Natural Science Foundation of China (NSFC) via grant 60675034, 60575042, and the "863" National High-Tech Research and Development of China via grant 2006AA01Z145.

References

Liu, Ting, Wanxiang Che, Sheng Li, Yuxuan Hu, and Huaijun Liu. 2005. Semantic role labeling system using maximum entropy classifier. In *Proceedings of CoNLL-2005*, June.

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

Punyakanok, Vasin, Dan Roth, Wen-tau Yih, and Dav Zimak. 2004. Semantic role labeling via integer linear programming inference. In *Proceedings of Coling-2004*, pages 1346–1352.

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)*.