Discriminative Learning of Syntactic and Semantic Dependencies

Lu Li[1], **Shixi Fan**[2], **Xuan Wang**[1], **Xiaolong Wang**[1]
Shenzhen Graduate School, Harbin Institute of Technology,
Xili, Shenzhen 518055, China
[1]{lli,wangxuan,wangxl}@insun.hit.edu.cn
[2]fanshixi@hit.edu.cn

Abstract

A Maximum Entropy Model based system for discriminative learning of syntactic and semantic dependencies submitted to the CoNLL-2008 shared task (Surdeanu, et al., 2008) is presented in this paper. The system converts the dependency learning task to classification issues and reconstructs the dependent relations based on classification results. Finally F1 scores of 86.69, 69.95 and 78.35 are obtained for syntactic dependencies, semantic dependencies and the whole system respectively in closed challenge. For open challenge the corresponding F1 scores are 86.69, 68.99 and 77.84.

1 Introduction

Given sentences and corresponding part-of-speech of each word, the learning of syntactic and semantic dependency contains two separable goals: (1) building a dependency tree that defines the syntactic dependency relationships between separated words; (2) specifying predicates (no matter verbs or nouns) of the sentences and labeling the semantic dependents for each predicate.

In this paper a discriminative parser is proposed to implement maximum entropy (ME) models (Berger, et al., 1996) to address the learning task. The system is divided into two main subsystems: syntactic dependency parsing and semantic dependency labeling. The former is used to find a well-formed syntactic dependency tree that occupies all the words in the sentence. If edges are added between any two words, a full-connected graph is constructed and the dependency tree could be found using a maximum spanning tree (MST) algorithm (McDonald, et al., 2005). The latter focuses on separable predicates whose semantic dependents could be determined using classification tools, such as ME models [1] etc..

We participated in both closed and open challenge of the CoNLL-2008 shared task (Surdeanu, et al., 2008). Results are reported on both development and test sets in this paper.

2 System Description

2.1 Syntactic Parsing

The goal of syntactic parsing is to create a labeled syntactic dependency parse **y** for input sentence **x** including words and their parts of speech (POS). Inspired by the parsing model that implements maximum spanning tree (MST) algorithm to induce the dependency parsing tree (McDonald, et al., 2005), the system employs the same framework. The incorporated features are defined over parts of speech of words occurring between and around a possible head-dependent relation.

Suppose $G = (V, E)$ is a directed graph, where V is the set of vertices denoting the words in sentence **x** and E is the set of directed edges between any two vertices with some scores. The MST algorithm is to find the most probable subgraph of G that satisfies tree constraints over all vertices. The score function of the parsing tree **y** is defined as

$$s(\mathbf{y}) = \sum_{(i,j)\in\mathbf{y}} s(i,j) \qquad (1)$$

where $(i, j) \in \mathbf{y}$ indicates an edge in **y** from word i to word j and $s(i, j)$ denotes its score. Suppose Y

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

CoNLL 2008: Proceedings of the 12th Conference on Computational Natural Language Learning, pages 218–222
Manchester, August 2008

w_i	w_j
p_i	p_j
(w_i, p_i)	(w_j, p_j)
(w_i, w_j)	(p_i, p_j)
(w_i, p_j)	(w_j, p_i)
(w_i, w_j, p_i)	(w_i, w_j, p_j)
(p_i, p_j, w_i)	(p_i, p_j, w_j)
(w_i, w_j, p_i, p_j)	$(p_i, p_k, p_j), i < k < j$
$(p_i, p_{i+1}, p_{j-1}, p_j)$	$(p_{i-1}, p_i, p_{j-1}, p_j)$
$(p_i, p_{i+1}, p_j, p_{j+1})$	$(p_{i-1}, p_i, p_j, p_{j+1})$

Table 1: Features for syntactic parsing.

w_i	p_i
p_{i-1}	p_{i+1}
(p_{i-1}, p_i)	(p_i, p_{i+1})
(p_{i-2}, p_i)	(p_i, p_{i+2})
(p_{i-3}, p_i)	(p_i, p_{i+3})
(p_{i-1}, p_i, p_{i+1})	(w_i, p_i)
(w_i, p_{i-1}, p_i)	(w_i, p_i, p_{i+1})
(w_i, p_{i-2}, p_i)	(w_i, p_i, p_{i+2})
(w_i, p_{i-3}, p_i)	(w_i, p_i, p_{i+3})
$(w_i, p_{i-1}, p_i, p_{i+1})$	

Table 2: Features used for predicate tagging.

is the set of syntactic dependency labels, the score function of edges is defined as

$$s(i, j) = max_{l \in \mathbf{Y}} Pr(l|\mathbf{x}, i, j) \quad (2)$$

ME models are used to calculate the value of $Pr(l|\mathbf{x}, i, j)$, where the features are extracted from input sentence **x**. Given i and j as the subscripts of words in the sentence and word i is the parent of word j, the features can be illustrated in table 1. w_i and p_i are denoted as the ith word and the ith part of speech respectively in the sentence. The tuples define integrated features, such as (w_i, p_i) indicates the feature combining the ith word and ith part of speech. Besides these features, the distant between word i and word j in sentence **x** is considered as a single feature. The distant is also combined with features in table 1 to produce complex features.

2.2 Semantic Dependency Labeling

Semantic dependencies are always concerning with specific predicates. Unlike syntactic dependencies, semantic dependency relationships usually can not be represented as a tree. Thus, the method used for semantic dependency labeling is somewhat different from syntactic dependency parsing. The work of semantic labeling can be divided into two stages: predicate tagging and dependents recognizing.

2.2.1 Predicate Tagging

According to PropBank (Palmer, et al., 2005) and NomBank (Meyers, et al., 2004), predicates usually have several rolesets corresponding to different meanings. For example, the verb *abandon* has three rolesets marked as ordinal numbers **01**, **02** and **03** as described below.

<frameset>
<predicate lemma="abandon">
<roleset id="abandon.**01**" name="leave behind" vncls="51.2">
...
</roleset>
<roleset id="abandon.**02**" name="exchange" vncls="51.2">
...
</roleset>
<roleset id="abandon.**03**" name="surrender, give_over" vncls="-">
...
</roleset>
</predicate>
</frameset>

The goal of this part is to identify the predicates in the sentences and to determine the roleset for each of them. It should be cleared that the ordinal numbers are only used to distinguish different meanings of a predicate. However, if these numbers are treated as tags for predicates, some statistical properties will be obtained as illustrated in Figure 1. As can be seen, the distribution of the *train* data would be quite informative for representing the distribution of other three data sets. Based on this idea, a classification framework is introduced for predicate tagging.

Suppose the tag set is chosen to be $T = \{\mathbf{01}, \mathbf{02}, ..., \mathbf{22}\}$ according to the horizontal axis of Figure 1 and **00** is added to indicate that the examining word is not a predicate. Suppose t_i is a variable indicating the tag of word at position i in sentences **x**. ME models are implemented to tag the predicates.

$$t_i = argmax_{t \in T} Pr(t|\mathbf{x}, i) \quad (3)$$

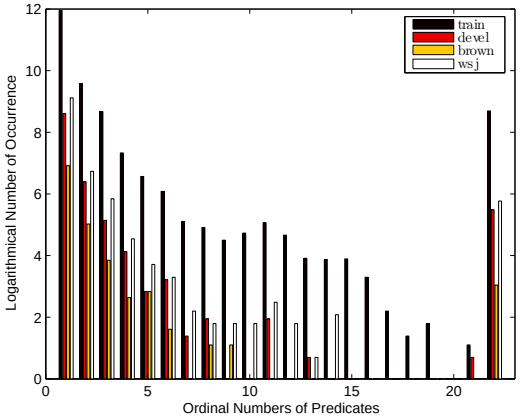

Figure 1: Distribution of the ordinal numbers of predicates on different data sets. 01 - 21 are attached with the predicates in the corpus and 22 stands for 'SU'.

The features for predicate tagging are listed in table 2, where the symbols share the same meaning as in table 1. Experiments show that this pure statistic processing method is effective for predicate tagging.

2.2.2 Dependents Recognizing

This subtask depends deeply on the results of syntactic parsing and predicate tagging described earlier in the system. Predicate tagging identifies central words and syntactic parsing provides syntactic features for its dependents identification and classification.

Generally speaking, given a specific predicate in a sentence, only a few of words are associated as its semantic dependents. By statistical analysis a list of part of speech tuples that are appearing to be semantic dependency are collected. All other tuples are filtered out to improve system performance.

Suppose (p, d) is a couple of predicate and one of its possible dependents, T is the dependency tree generated by syntactic parsing, L is the set of semantic dependency labels. The dependents can be recognized by using a classification model, ME models are chosen as before.

$$l_{(p,d)} = argmax_{l \in L} Pr(l|p, d, T) \qquad (4)$$

Besides the semantic dependency labels, *null* is included as a special tag to indicate that there is no semantic dependency between p and d. As a result, dependents identification (binary classification) and dependents tagging (multi-classification) can be solved together within one multi-classification framework.

The selected features are listed below.

1. Predicate Features
   - **Lemma and POS** of predicate, predicate's parent in syntactic dependency tree.
   - **Voice** active or passive.
   - **Syntactic dependency label** of edge between predicate and its parent.
   - **POS framework** POS list of predicate's siblings, POS list of predicate's children.
   - **Syntactic dependency framework** syntactic dependency label list of the edges between predicate's parent and its siblings.
   - **Parent framework** syntactic dependency label list of edges connecting to predicate's parent.
2. Dependent Features
   - **Lemma and POS** of dependent, dependent's parent.
   - **POS framework** POS list of dependent's siblings.
   - **Number of children** of dependent's parent.
3. In Between Features
   - **Position** of dependent according to predicate: before or after.
   - **POS pair** of predicate and dependent.
   - **Family relation** between predicate and dependent: ancestor or descendant.
   - **Path length** between predicate and dependent.
   - **Path POS** POS list of all words appearing on the path from predicate to dependent.
   - **Path syntactic dependency label** list of dependency label of edges of path between predicate and dependent.

3 Experiment results

The classification models were trained using all the training data. The detailed information are shown in table 3. All experiments ran on 32-bit Intel(R) Pentium(R) D CPU 3.00GHz processors with 2.0G memory.

	Feature Number	Training Time
Syn.	7,488,533	30h
Prd.	1,484,398	8h
Sem.	3,588,514	12h

Table 3: Details of ME models. *Syn.* is for syntactic parsing, *Prd.* is for predicate tagging and *Sem.* is for semantic dependents recognizing.

	Syntactic	Semantic	Overall
devel	85.29	69.60	77.49
brown	80.80	59.17	70.01
wsj	87.42	71.27	79.38
brown+wsj	86.69	69.95	78.35

(a) Closed Challenge

	Syntactic	Semantic	Overall
devel	85.29	68.45	76.87
brown	80.80	58.22	69.51
wsj	87.42	70.32	78.87
brown+wsj	86.69	68.99	77.84

(b) Open Challenge

Table 4: Scores for joint learning of syntactic and semantic dependencies.

3.1 Closed Challenge

The system for closed challenge is designed as a two-stage parser: syntactic parsing and semantic dependency labeling as described previously. Table 4(a) shows the results on different corpus. As shown in table 4(a), the scores of semantic dependency labeling are quite low, that are influencing the overall scores. The reason could be inferred from the description in section 2.2.2 since semantic dependent labeling inherits the errors from the output of syntactic parsing and predicate tagging. Following evaluates each part independently.

Besides the multiple classification model described in table 3, a binary classification model was built based on ME for predicate tagging. The binary model can't distinguish different rolesets of predicate, but can identify which words are predicates in sentences. The precision and recall for binary model are *90.80* and *88.87* respectively, while for multiple model, the values are *84.60* and *85.60*.

For semantic dependent labeling, experiments were performed under conditions that the gold syntactic dependency tree and predicates list were given as input. The semantic scores became *80.09*, *77.08* and *82.25* for *devel*, *brown* and *wsj* respectively. This implies that the error of syntactic parsing and predicate tagging could be probably augmented in semantic dependent labeling. In order to improve the performance of the whole system, the deep dependence between the two stages should be broken up in future research.

3.2 Open Challenge

In open challenge, the same models are used for syntactic parsing and predicate tagging as in closed challenge and two other models are trained for semantic dependent labeling. Suppose M_{mst}, M_{malt} and M_{chunk} are denoted as these three semantic models, where M_{mst} is the model used in closed challenge, M_{malt} is trained on the syntactic dependency tree provided by the open corpus with the same feature set as M_{mst}, and M_{chunk} is trained using features extracted from name entity and wordnet super senses results provided by the open corpus.

Considering a possible dependent given a specific predicate, the feature set used for M_{chunk} contains only six elements:

- Whether the dependent is in name entity chunk: True or False.
- Name entity label of the dependent.
- Whether the dependent is in BBN name entity chunk: True or False.
- BBN name entity label of the dependent.
- Whether the dependent is in wordnet super sense chunk: True or False.
- Wordnet super sense label of the dependent.

After implementing these three models on semantic dependents recognizing, the results were merged to generate the scores described in table 4(b).

The merging strategy is quite simple. Given a couple of predicate and dependent (p, d), the system produces three semantic dependency labels denoting as l_{mst}, l_{malt} and l_{chunk}, the result label is chosen to be most frequent semantic label among the three.

Comparing the scores of open challenge and closed challenge, it can be found that the score of the former is less than the latter, which is quite strange since more resources were used in open challenge. To examine the influences of different semantic dependents recognizing models, each

	M_{mst}	M_{malt}	M_{chunk}
devel	69.60	64.48	41.72
brown	59.17	56.52	34.04
wsj	71.27	66.40	41.83

Table 5: Semantic scores of different models.

model was implemented in the closed challenge and the results are shown in table 5. Specially, model M_{chunk} generated too low scores and gave a heavy negative influence on the final results. Finding a good way to combine several results requires further research.

4 Conclusions

This paper have presented a simple discriminative system submitted to the CoNLL-2008 shared task to address the learning task of syntactic and semantic dependencies. The system was divided into syntactic parsing and semantic dependents labeling. Maximum spanning tree was used to find a syntactic dependency tree in the full-connected graph constructed over the words of a sentence. Maximum entropy models were implemented to classify syntactic dependency edges, predicates and their semantic dependents. A brief analysis has also been given on the results of both closed challenge and open challenge.

Acknowledgement

This research has been partially supported by the National Natural Science Foundation of China (No. 60435020 and No. 90612005), the Goal-oriented Lessons from the National 863 Program of China (No.2006AA01Z197) and Project of Microsoft Research Asia. We would like to thank Zhixin Hao, Xiao Xin, Languang He and Tao Qian for their wise suggestion and great help. Thanks also to Muhammad Waqas Anwar for English improvement.

References

Adam Berger, Stephen Della Pietra, Vincent Della Pietra 1996. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, 22(1):39-71.

Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young and Ralph Grishman 2004. The NomBank Project: An Interim Report *HLT-NAACL 2004 Workshop: Frontiers in Corpus Annotation*, 24-31.

Martha Palmer, Daniel Gildea, Paul Kingsbury 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles *Computational Linguistics*, 31(1):71-106.

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. *Proceedings of the 12th Conference on Computational Natural Language Learning (CoNLL-2008)*

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič 2005. Non-projective Dependency Parsing using Spanning Tree Algorithms. *Proceedings of HLT/EMNLP*, 523-530.