Parsing Syntactic and Semantic Dependencies with Two Single-Stage Maximum Entropy Models *

Hai Zhao and Chunyu Kit
Department of Chinese, Translation and Linguistics
City University of Hong Kong
83 Tat Chee Avenue, Kowloon, Hong Kong, China
haizhao@cityu.edu.hk, ctckit@cityu.edu.hk

Abstract

This paper describes our system to carry out the joint parsing of syntactic and semantic dependencies for our participation in the shared task of CoNLL-2008. We illustrate that both syntactic parsing and semantic parsing can be transformed into a word-pair classification problem and implemented as a single-stage system with the aid of maximum entropy modeling. Our system ranks the fourth in the closed track for the task with the following performance on the WSJ+Brown test set: 81.44% labeled macro F1 for the overall task, 86.66% labeled attachment for syntactic dependencies, and 76.16% labeled F1 for semantic dependencies.

1 Introduction

The joint parsing of syntactic and semantic dependencies introduced by the shared task of CoNLL-08 is more complicated than syntactic dependency parsing or semantic role labeling alone (Surdeanu et al., 2008). For semantic parsing, in particular, a dependency-based representation is given but the predicates involved are unknown, and we also have nominal predicates besides the verbal ones. All these bring about more difficulties for learning. This paper presents our research for participation in the CoNLL-2008 shared task, with a highlight on our strategy to select learning framework and features for maximum entropy learning.

The rest of the paper is organized as follows. The next section presents the technical details of our system and Section 3 its evaluation results. Section 4 looks into a few issues concerning our forthcoming work for this shared task, and Section 5 concludes the paper.

2 System Description

For the sake of efficiency, we opt for the maximum entropy model with Gaussian prior as our learning model for both the syntactic and semantic dependency parsing. Our implementation of the model adopts L-BFGS algorithm for parameter optimization as usual (Liu and Nocedal, 1989). No additional feature selection techniques are applied.

Our system consists of three components to deal with syntactic and semantic dependency parsing and word sense determination, respectively. Both parsing is formulated as a single-stage word-pair classification problem, and the latter is carried out by a search through the NomBank (Meyers et al., 2004) or the PropBank (Palmer et al., 2005)[1].

2.1 Syntactic Dependency Parsing

We use a shift-reduce scheme to implement syntactic dependency parsing as in (Nivre, 2003). It takes a step-wised, history- or transition-based approach. It is basically a word-by-word method with a projective constraint. In each step, the classifier checks a word pair, e.g., *TOP*, the top of a stack for processed words, and, *NEXT*, the first word in the unprocessed word sequence, in order to determine if a dependent label should be assigned to them. Besides two arc-building actions, a shift action and a reduce action are also defined to meet the projective constraint, as follows.

This study is supported by CERG grant 9040861 (CityU 1318/03H) and CityU Strategic Research Grant 7002037.

[1] These two dictionaries that we used are downloaded from CoNLL-2008 official website.

Notation	Meaning
s	Clique in the top of stack
s_{-1},...	The first clique below the top of stack, etc.
i, i_{+1},...	The first (second) clique in the unprocessed sequence, etc.
dprel	Dependent label
h	Head
lm	Leftmost child
rm	Rightmost child
rn	Right nearest child
form	Word form
lemma	Word lemma
pos	Predicted PoS tag
sp_Y	Split Y, which may be *form*, *lemma* or *pos*.
.	's, e.g., '*s.dprel*' means dependent label of the clique in the top of stack
/	Feature combination, i.e., '*s.pos/i.pos*' means *s.pos* and *i.pos* together as a feature function.
p	The current predicate candidate
a	The current argument candidate

Table 1: Feature Notations

Basic	Extension
$x.sp\_Y$	itself, its previous two and next two Ys, and all bigrams within the five-clique window, (x is s or i, and Y is *form*, *lemma* or *pos*.)
$x.Y$	(x is s or i, and Y is *form*, *lemma* or *pos*.)
$x.Y/i.Y$	(x is s or s_{-1} and Y is *pos*, *sp_lemma* or *sp_pos*)
_	*s.h.sp_form*
_	*s.dprel*
_	*s.lm.dprel*
_	*s.rn.dprel*
_	*i.lm.sp_pos*
_	*s.lm.dprel/s.dprel*
_	*s.lm.sp_pos/s.sp_pos*
_	*s.h.sp_pos/s.sp_pos*
_	*x.sp_pos*|*rootscore* (x is s or i.)
_	*s.sp_pos/i.sp_pos*|*pairscore*
_	*s.curroot.sp_pos/i.sp_pos*

Table 2: Features for Syntactic Parsing

(1) **Left-arc**: Add an arc from *NEXT* to *TOP* and pop the stack.

(2) **Right-arc**: Add an arc from *TOP* to *NEXT* and push *NEXT* onto the stack.

(3) **Reduce**: Pop *TOP* from the stack.

(4) **Shift**: Push *NEXT* onto the stack.

We implement a left-to-right arc-eager parsing model in a way that the parser scan through an input sequence from left to right and the right dependents are attached to their heads as soon as possible (Hall et al., 2007). To construct a single-stage system, we extend the left-/right-arc actions to their correspondent multi-label actions as necessary. Including 32 left-arc and 66 right-arc actions, altogether a 100-class problem is yielded for the parsing action classification for this shared task.

Since only projective sequences can be handled by the shift-reduce scheme, we apply the pseudo-projective transformation introduced by (Nivre and Nilsson, 2005) to projectivize those non-projective sequences. Our statistics show that only 7.6% sequences and less than 1% dependencies in the corpus provided for training are non-projective. Thus, we use a simplified strategy to projectivize an input sequence. Firstly, we simply replace the head of a non-projective dependency by its original head's head but without any additional dependent label encoding for the purpose of deprojectivizing the output during decoding. Secondly, if the above standard projectivization step cannot eliminate all non-projective dependencies in a sequence, then the word with the shortest sequence (rather than dependent tree) distance to the original head will be chosen as the head of a non-projective dependency. In practice, the above two-step projectivization procedure can eliminate all non-projective dependencies in all sequences. Our purpose here is to provide as much data as possible for training, and only projective sequences are input for training and output for decoding.

While memory-based and margin-based learning approaches such as support vector machines are popularly applied to shift-reduce parsing, our work provides evidence that the maximum entropy model can achieve a comparative performance with the aid of a suitable feature set. With feature notations in Table 1, we use a feature set as shown in Table 2 for syntactic parsing.

Here, we explain '*rootscore*', '*pairscore*' and *curroot* in Table 2. Both *rootscore* and *pairscore* return the log frequency for an event in the training corpus. The former counts a given split PoS occurring as ROOT, and the latter two split PoS's combination associated with a dependency label. The feature *curroot* returns the root of a partial parsing tree that includes a specified node.

2.2 Semantic Dependency Parsing

Assuming no predicates overtly known, we keep using a word-pair classifier to perform semantic parsing through a single-stage processing. Specifically, we specify the first word in a word pair as a predicate candidate (i.e., a semantic head, and noted as p in our feature representation) and the

Basic	Extension
x.sp_Y	itself, its previous and next cliques, and all bigrams within the three-clique window. (Y is *form* or *lemma*.)[a]
x.sp_pos	itself, its previous and next two cliques, and all bigrams within the five-clique window.
$x.Y$	(Y is *form*, *lemma* or *pos*.)
$p.Y/i.Y$	(Y is *sp_lemma* or *sp_pos*.)
_	a is the same as p
_	*x.is_Verb_or_Noun*
_	*bankAdvice*
_[b]	*a.h.sp_form*
_	*x.dprel*
_	*x.lm.dprel*
_	*p.rm.dprel*
_	*p.lm.sp_pos*
_	*a.lm.dprel/a.dprel*
_	*a.lm.sp_pos/a.sp_pos*
_	*a.sp_Y/a.dprel* (Y is *lemma* or *pos*.)
_	*x.sp_lemma/x.h.sp_form*
_	*p.sp_lemma/p.h.sp_pos*
_	*p.sp_pos/p.dprel*
_	*a.preddir*[c]
_	*p.voice/a.preddir*[d]
_	*x.posSeq*[e]
_	*x.dprelSeq*[f]
_	*a.dpTreeLevel*[g]
_	a/p|*dpRelation*
_	a/p|*SharedPosPath*
_	a/p|*SharedDprelPath*
_	a/p|*x.posPath*
_	a/p|*x.dprelPath*
_	a/p|*dprelPath*

[a] x is p or a throughout the whole table.

[b] This and the following features are all concerned with a known syntactic dependency tree.

[c] *preddir*: the direction to the current predicate candidate.

[d] *voice*: if the syntactic head of p is *be* and p is not ended with *-ing*, then p is passive.

[e] *posSeq*: PoS tag sequence of all syntactic children

[f] *dprelSeq*: syntactic dependent label sequence of all syntactic children

[g] *dpTreeLevel*: the level in the syntactic parse tree, counted from the leaf node.

Table 3: Features for Semantic Parsing

next as an argument candidate (i.e., a semantic dependent, and noted as a). We do not differentiate between nominal and verbal predicates and our system handles them in in exactly the same way. If decoding outputs show that no arguments can be found for a predicate candidate in the decoding output, then this candidate will be naturally discarded from the output predicate list.

When no constraint available, however, all word pairs in the an input sequence must be considered, leading to very poor efficiency in computation for no gain in effectiveness. Thus, the training sample needs to be pruned properly.

For predicate, only nouns and verbs are considered possible candidates. That is, all words without a split PoS in these two categories are filtered out. Many prepositions are also marked as predicate in the training corpus, but their arguments' roles are '*SU*', which are not counted the official evaluation.

For argument, a dependency version of the pruning algorithm in (Xue and Palmer, 2004) is used to find, in an iterative way, the current syntactic head and its siblings in a parse tree in a constituent-based representation. In this representation, the head of a phrase governs all its sisters in the tree, as illustrated in the conversion of constituents to dependencies in (Lin, 1995). In our implementation, the following equivalent algorithm is applied to select argument candidates from a syntactic dependency parse tree.

> Initialization: Set the given predicate candidate as the current node;
>
> (1) The current node and all of its syntactic children are selected as argument candidates.
>
> (2) Reset the current node to its syntactic head and repeat step (1) until the root is reached.

This algorithm can cover 98.5% arguments while reducing about 60% of the training samples, according to our statistics. However, this is achieved at the price of including a syntactic parse tree as part of the input for semantic parsing.

The feature set listed in Table 3 is adopted for our semantic parsing, some of which are borrowed from (Hacioglu, 2004). Among them, *dpTreeRelation* returns the relationship of a and p in a syntactic parse tree. Its possible values include `parent`, `sibling`, `child`, `uncle`, `grand parent` etc. Note that there is always a path to the ROOT in the syntactic parse tree for either a or p. Along the common part of these two paths, *SharedDprelPath* returns the sequence of dependent labels collected from each node, and *SharedPosPath* returns the corresponding sequence of PoS tags. x.*dprelPath* and x.*posPath* return the PoS tag sequence from x to the beginnings of *SharedDprelPath* and *SharedPosPath*, respectively. a/p|*dprelPath* returns the concatenation of a.*dprelPath* and p.*dprelPath*.

We may have an example to show how the feature *bankAdvice* works. Firstly, the current processed semantic role labels and argument candidate direction are checked. Specifically, they are

	UAS	LAS	Label-Acc.
Development	88.78	85.85	91.14
WSJ	89.86	87.52	92.47
Brown	85.03	79.83	86.71
WSJ+Brown	89.32	86.66	91.83

Table 4: The Results of Syntactic Parsing (%)

	Data	Precision	Recall	F-score
Label.	Development	79.76	72.25	75.82
	WSJ	80.57	74.97	77.67
	Brown	66.28	61.29	63.69
	WSJ+Brown	79.03	73.49	76.16
Unlab.	Development	89.58	81.15	85.16
	WSJ	89.48	83.26	86.26
	Brown	83.14	76.88	79.89
	WSJ+Brown	88.79	82.57	85.57

Table 5: The Results of Semantic Parsing (%)

	Data	Precision	Recall	F-score
Label. Macro	Development	82.80	79.05	80.88
	WSJ	84.05	81.25	82.62
	Brown	73.05	70.56	71.78
	WSJ+Brown	82.85	80.08	81.44
Unlab. Macro	Development	89.18	84.97	87.02
	WSJ	89.67	86.56	88.09
	Brown	84.08	80.96	82.49
	WSJ+Brown	89.06	85.94	87.47
Label. Micro	Development	83.69	80.71	82.17
	WSJ	85.07	82.88	83.96
	Brown	75.14	73.09	74.10
	WSJ+Brown	83.98	81.80	82.88
Unlab. Micro	Development	89.06	85.90	87.45
	WSJ	89.72	87.42	88.56
	Brown	84.38	82.07	83.21
	WSJ+Brown	89.14	86.83	87.97

Table 6: Overall Scores (%)

the arguments $A0$ and $A1$ that have been marked before the predicate candidate p and the current argument identification direction after p. Secondly, each example[2] of p in NomBank or PropBank that depends on the split PoS tag of p is checked if it partially matches the current processed role labels. If a unique example exists in this form, e.g., `Before:`$A0$–$A1$`; After:`$A3$, then this feature returns $A3$ as feature value. If no matched or multiple matched examples exist, then this feature returns a default value.

2.3 Word Sense Determination

The shared task of CoNLL-2008 for word sense disambiguation task is to determine the sense of an output predicate. Our system carries out this task by searching for a right example in the given NomBank or PropBank. The semantic role set scheme of each example for an output predicate is checked. If a scheme is found to match the output semantic role set of a predicate, then the corresponding sense for the first match is chosen; otherwise the system outputs '`01`' as the default sense.

3 Evaluation Results

Our evaluation is carried out on a 64-bit ubuntu Linux installed server with double dual-core AMD Opteron processors of 2.8GHz and 8GB memory. The full training set for CoNLL-2008 is used to train the maximum entropy model. The training for the syntactic parser costs about 200 hours and 4.1GB memory and that for the semantic parser costs about 170 hours and 4.9GB memory. The running time in each case is the sum of all running time for all threads involved. When a parallel optimization technique is applied to speedup the training, the time can be reduced to about 1/3.5 of the above.

The official evaluation results for our system are presented in Tables 4, 5 and 6. Following the official guideline of CoNLL-2008, we use unlabeled attachment score (UAS), labeled attachment score (LAS) and label accuracy to assess the performance of syntactic dependency parsing. For semantic parsing, the unlabeled scores metric the identification performance and the labeled scores the overall performance of semantic labeling.

4 To Do

Although we are unable to follow our plan to do more than what we have done for this shared task, because of the inadequate computational resource and limited time, we have a number of techniques in our anticipation to bring in further performance improvement.

While expecting to accomplish the joint inference of syntactic and semantic parsing, we only have time to complete a system with the former to enhance the latter. But we did have experiments in the early stage of our work to show that a syntactic dependency parser can make use of avail-

[2]The term "example" means a chunk in NomBank or PropBank, which demonstrates how semantic roles occur around a specified predicate. For example, for a sense item of the predicate `access` in PropBank, we first have `<arg n="0">a computer</arg> <rel>access</rel> <arg n="1">its memory</arg>`, and then a role set scheme for this sense as `Before:`$A0$`;After:`$A1$.

able semantic dependency information to enhance its performance by 0.5-1% [3].

Most errors in our syntactic parsing are related to the dependencies of comma and prepositions. We need to take care of them, for PP attachment is also crucial to the success of semantic parsing. Extra effort is paid, as illustrated in previous work such as (Xue and Palmer, 2004), to handle such cases, especially when a PP is involved. We find in our data that about 1% arguments occur as a grandchild of a predicate through PP attachment.

Syntactic parsing contributes crucially to the overall performance of the joint parsing by providing a solid basis for further semantic parsing. Thus there is reason to believe that improvement of syntactic dependency parsing can be more influential than that of semantic parsing to the overall improvement. Only one model was used for syntactic parsing in our system, in contrast to the existing work using an ensemble technique for further performance enhancement, e.g., (Hall et al., 2007). Again, the latter means much more computational cost should be taken.

Though it was not done before submission deadline, we also tried to enhance the semantic parsing with some more sophisticated inputs from the syntactic parsing. One is predicted syntactic parsed tree input that may be created by cross-validation rather than the gold-standard syntactic input that our submitted semantic parser was actually trained on. Another is the n-best outputs of the syntactic parser. However, only the single-best output of the syntactic parser was actually used.

5 Conclusion

As presented in the above sections, our system to participate in the CoNLL-2008 shared task is implemented as two single-stage maximum entropy learning. We have tackled both syntactic and semantic parsing as a word-pair classification problem. Despite the simplicity of this approach, our system has produced promising results.

Acknowledgements

We wish to thank Dr. Wenliang Chen of NICT, Japan for helpful discussions on dependency parsing, and two anonymous reviewers for their valuable comments.

[3] We used the outputs of a semantic parser, either predicted or gold-standard, as features for syntactic parsing.

References

Hacioglu, Kadri. 2004. Semantic role labeling using dependency trees. In *Proceedings of the 20th international conference on Computational Linguistics (COLING-2004)*, pages 1273–1276, Geneva, Switzerland, August 23rd-27th.

Hall, Johan, Jens Nilsson, Joakim Nivre, Gülsen Eryiğit, Beáta Megyesi, Mattias Nilsson, and Markus Saers. 2007. Single malt or blended? a study in multilingual parser optimization. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 933–939, Prague, Czech, June.

Lin, Dekang. 1995. A dependency-based method for evaluating broad-coverage parser. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 1420–1425, Montréal, Québec, Canada, August 20-25.

Liu, Dong C. and Jorge Nocedal. 1989. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45:503–528.

Meyers, Adam, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. 2004. The nombank project: An interim report. In *Proceedings of HLT/NAACL Workshop on Frontiers in Corpus Annotation*, pages 24–31, Boston, Massachusetts, USA, May 6.

Nivre, Joakim and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics (ACL-2005)*, pages 99–106, Ann Arbor, Michigan, USA, June 25-30.

Nivre, Joakim. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT 03)*, pages 149–160, Nancy, France, April 23-25.

Palmer, Martha, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

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)*.

Xue, Nianwen and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *2004 Conference on Empirical Methods in Natural Language Processing (EMNLP-2004)*, pages 88–94, Barcelona, Spain, July 25-26.