Hybrid Learning of Dependency Structures from Heterogeneous Linguistic Resources

Yi Zhang
Language Technology Lab
DFKI GmbH
yzhang@coli.uni-sb.de

Rui Wang
Computational Linguistics
Saarland University, Germany
rwang@coli.uni-sb.de

Hans Uszkoreit
Language Technology Lab
DFKI GmbH
uszkoreit@dfki.de

Abstract

In this paper we present our syntactic and semantic dependency parsing system participated in both closed and open competitions of the CoNLL 2008 Shared Task. By combining the outcome of two state-of-the-art syntactic dependency parsers, we achieved high accuracy in syntactic dependencies (87.32%). With MRSes from grammar-based HPSG parsers, we achieved significant performance improvement on semantic role labeling (from 71.31% to 71.89%), especially in the out-domain evaluation (from 60.16% to 62.11%).

1 Introduction

The CoNLL 2008 shared task (Surdeanu et al., 2008) provides a unique chance of comparing different syntactic and semantic parsing techniques in one unified open competition. Our contribution in this joint exercise focuses on the combination of different algorithms and resources, aiming not only for state-of-the-art performance in the competition, but also for the dissemination of the learnt lessons to related sub-fields in computational linguistics.

The so-called hybrid approach we take has two folds of meaning. For syntactic dependency parsing, we build our system based on state-of-the art algorithms. Past CoNLL share task results have shown that transition-based and graph-based algorithms started from radically different ideas, yet achieved largely comparable results. One of the question we would like investigate is whether the combination of the two approach on the output level leads to even better results.

For the semantic role labeling (SRL) task, we would like to build a system that allows us to test the contribution of different linguistic resources. To our special interest is to examine the deep linguistic parsing systems based on hand-crafted grammars. During the past decades, various large scale linguistic grammars have been built, some of which achieved both broad coverage and high precision. In combination with other advances in deep linguistic processing, e.g. efficient parsing algorithms, statistical disambiguation models and robust processing techniques, several systems have reached mature stage to be deployed in applications. Unfortunately, due to the difficulties in cross-framework evaluation, fair comparison of these systems with state-of-the-art data-driven statistical parsers is still hard to achieve. More importantly, it is not even clear whether deep linguistic analysis is necessary at all for tasks such as shallow semantic parsing (also known as SRL). Drawing a conclusion on this latter point with experiments using latest deep parsing techniques is one of our objective.

The remainder of the paper is structure as follows. Section 2 introduces the overall system architecture. Section 3 explains the voting mechanism used in the syntactic parser. Section 4 describes in detail the semantic role labeling component. Section 5 presents evaluation results and error analysis. Section 6 concludes the paper.

2 System Architecture

As shown in Figure 1, our system is a two-stage pipeline. For the syntactic dependencies, we apply two state-of-the-art dependency parsers and combined their results based on a voting model. For

CoNLL 2008: Proceedings of the 12th Conference on Computational Natural Language Learning, pages 198–202
Manchester, August 2008

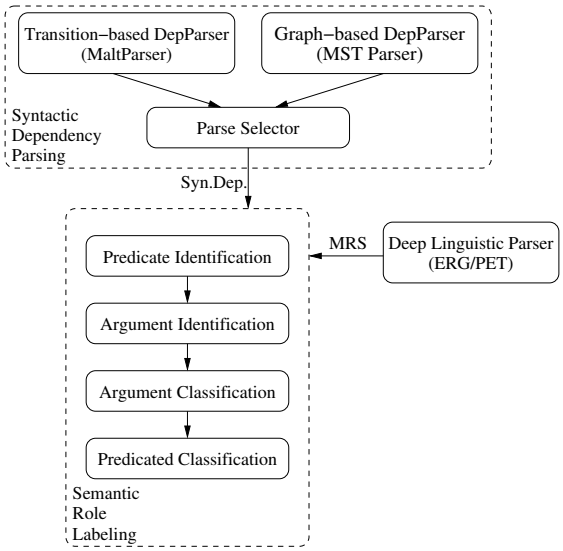

Figure 1: System Architecture

the semantic roles, we extracted features from the previous stage, combined with deep parsing results (in MRS), and use statistical classification models to make predictions. In particular, the second part can be further divided into four stages: predicate identification (PI), argument identification (AI), argument classification (AC), and predicate classification (PC). Maximum entropy-based machine learning techniques are used in both components which we will see in detail in the following sections.

3 Syntactic Dependency Parsing

For obtaining syntactic dependencies, we have combined the results of two state-of-the-art dependency parsers: the MST parser (McDonald et al., 2005) and the MaltParser (Nivre et al., 2007).

The MST parser formalizes dependency parsing as searching for maximum spanning trees (MSTs) in directed graphs. A major advantage of their framework is the ability to naturally and efficiently model both projective and non-projective parses. To learn these structures they used online large-margin learning that empirically provides state-of-the-art performance.

The MaltParser is a transition-based incremental dependency parser, which is language-independent and data-driven. It contains a deterministic algorithm, which can be viewed as a variant of the basic shift-reduce algorithm. The learning method they applied is support vector machine and experimental evaluation confirms that the MaltParser can achieve robust, efficient and accurate parsing for a wide range of languages.

Since both their parsing algorithms and machine learning methods are quite different, we decide to take advantages of them. After a comparison between the results of the two parsers[1], we find that,

1. The MST parser is better at the whole structure. In several sentences, the MaltParser was wrong at the root node, but the MST parser is correct.

2. The MaltParser is better at some dependency labels (e.g. TMP, LOC, etc.).

These findings motivate us to do a voting based on both outputs. The features considered in the voting model are as follows:

- **Dependency path**: two categories of dependency paths are considered as features: 1) the POS-Dep-POS style and 2) the Dep-Dep style. The former consists of part-of-speech (POS) tags and dependency relations appearing in turns; and the latter only contains dependency relations. The maximum length of the dependency path is three dependency relations.

- **Root attachments**: the number of tokens attached to the ROOT node by the parser in one sentence

- **Sentence length**: the number of tokens in each input sentence

- **Projectivity**: whether the parse is projective or not

With these features, we apply a statistical model to predict, for each sentence, we choose the parsing result from which parser. The voted result will be our syntactic dependency output and be passed to the later stages.

4 Semantic Role Labeling

4.1 Overview

The semantic role labeling component of our system is comprised of a pipeline model with four

[1]In this experiment, we use second order features and the projective decoder for the MST parser trained with 10 iterations, and Arc-eager algorithm with a quadric polynomial kernel for the MaltParser.

sub-components that performs predicate identification (PI), argument identification (AI), argument classification (AC) and predicate classification (PC) respectively. The output in previous steps are taken as input information to the following stages. All these components are essentially based on a maximum entropy statistical classifier, although with different task-specific optimizations and feature configurations in each step. Depending on the available information from the input data structure, the same architecture is used for both closed and open challenge runs, with different feature types. Note that our system does not make use of or predict SU chains.

Predicate Identification The component makes binary prediction on each input token whether it forms a predicate in the input sentence. This predictor precedes other components because it is a relatively easy task (comparing to the following components). Also, making this prediction early helps to cut down the search space in the following steps. Based on the observation on the training data, we limit the PI predictor to only predict for tokens with certain POS types (POSes marked as predicates for at least 50 times in the training set). This helps to significantly improve the system efficiency in both training and prediction time without sacrificing prediction accuracy.

It should be noted that the prediction of nominal predicates are generally much more difficult (based on CoNLL 2008 shared task annotation). The PI model achieved 96.32 F-score on WSJ with verbal predicates, but only 84.74 on nominal ones.

Argument Identification After PI, the arguments to the predicted predicates are identified with the AI component. Similar to the approach taken in Hacioglu (2004), we use a statistical classifier to select from a set of candidate nodes in a dependency tree. However, instead of selecting from a set of neighboring nodes from the predicate word [2], we define the concept of argument path as a chain of dependency relations from the predicate to the argument in the dependency tree. For instance, an argument path [SBJ | TMP] indicates that if the predicate is syntactically depending as SBJ on a node which has a TMP child, then the TMP node (sibling to the predicate) is an argument candidate. While Hacioglu (2004)'s approach focus mainly on local arguments (with respect to the syntactic dependencies), our approach is more suitable of capturing long distance arguments from the predicate. Another minor difference is that we allow predicate word to be its own argument (which is frequently the case for nominal predicates) with an empty argument path [|].

The set of effective argument paths are obtained from the training set, sorted and filtered according to their frequencies, and used in testing to obtain the candidate arguments. By setting a frequency threshold, we are able to select the most useful argument paths. The lower the threshold is, the higher coverage one might get in finding candidate arguments, accompanied with a higher average candidate number per predicate and potentially a more difficult task for the statistical classifier. By experimenting with different frequency thresholds on the training set, we established a frequency threshold of 40, which guarantees candidate argument coverage of 95%, and on average 5.76 candidates per predicate. Given that for the training set each predicate takes on average 2.13 arguments, the binary classifier will have relatively balanced prediction classes.

Argument Classification For each identified argument, an argument label will be assigned during the argument classification step. Unlike the binary classifiers in previous two steps, AC uses a multi-class classifier that predicts from the inventory of argument labels. For efficiency reasons, we only concern the most frequent 25 argument labels.

Predicate Classification The final step in the SRL component labels the predicted predicate with a predicate name. Due to the lack of lexical resources in the closed competition, this step is scheduled for the last, in order to benefit from the predictions made in the previous steps. Unlike the previous steps, the statistical model used in this step is a ranking model. We obtained a list of candidate frames and corresponding rolesets from the provided PropBank and NomBank data. Each predicted predicate is mapped onto the potential rolesets it may take. When the frame for the predicate word is missing from the list, or there is only one candidate roleset for it, the predicate name is assigned deterministically (word stem concatenated with ".01" for frame missing predicates, the unam-

[2] Hacioglu (2004) defines a tree-structured *family* of a predicate as a measure of locality. It is a set of dependency relation nodes that consists of the predicate's parent, children, grandchildren, siblings, siblings' children and siblings' grandchildren with respect to its dependency tree

biguous roleset name when there is only one candidate). When there are more than one candidate rolesets, a ranking model is trained to select the most probable roleset for a given predicate given the syntactic and semantic context.

4.2 Features

The feature types used in our SRL component are summarized in Table 1, with the configurations of our submitted "closed" and "open" runs marked. Numerous different configurations with these feature types have been experimented on training and development data. The results show that feature types 1–14 are the best performing ones. Features related to the siblings of the predicate only introduce minor performance variation. We also find the named entity labels does not lead to immediate improvement of SRL performance. The WordNet sense feature does achieve minor performance increase on PI and PC, although the significant remains to be further examined. Based on the pipeline model, we find it difficult to achieve further improvement by incorporate more features types from provided annotation. And the variance of SRL performance with different open features is usually less than 1%. To clearly show the contribution of extra external resources, these less contributing features (siblings, named entity labels and WordNet sense) are not used in our submitted "open" runs.

MRSes as features to SRL As a novel point of our SRL system, we incorporate parsing results from a linguistic grammar-based parsing system in our "open" competition run. In this experiment, we used English Resource Grammar (ERG, Flickinger (2000)), a precision HPSG grammar for English. For parsing, we used PET (Callmeier, 2001), an efficient HPSG parser, together with extended partial parsing module (Zhang et al., 2007) for maximum robustness. The grammar is handcrafted and the disambiguation models are trained on Redwoods treebanks. They present general linguistic knowledge, and are not tuned for the specific domains in this competition.

While the syntactic analysis of the HPSG grammar is largely different from the dependency annotation used in this shared task, the semantic representations do share a similar view on predicate-argument structures. ERG uses as its semantic representation the Minimal Recursion Semantics (MRS, Copestake et al. (2005)), a non-recursive flat structure that is suitable for underspecifying scope ambiguities. A predicate-argument backbone of MRS can be extracted by identifying shared variables between elementary predications (EPS).

In order to align the HPSG parser's I/O with CoNLL's annotation, extensive mapping scripts are developed to preprocess the texts, and extract backbone from output MRSes. The unknown word handling techniques (Zhang and Kordoni, 2005) are used to overcome lexical gaps. Only the best analysis is used for MRS extraction. Without partial parsing, the ERG achieves around 70% of raw coverage on the training data. When partial parsing is used, almost all the sentences received either full or partial analysis (except for several cases where computational resources are exhausted), and the SRL performance improves by ~0.5%.

5 Results

Among 20+ participant groups, our system ranked seventh in the "closed" competition, and first in the "open" challenge. The performance of the syntactic and semantic components of our system are summarized in Table 2.

		In-Domain		Out-Domain	
		Lab.	Unlab.	Lab.	Unlab.
Syntactic Dep.		88.14%	90.78%	80.80%	86.12%
SRL	Closed	72.67%	82.68%	60.16%	76.98%
	Open	73.08%	83.04%	62.11%	78.48%

Table 2: Labeled and unlabeled attachment scores in syntactic dependency parsing and F1 score for semantic role labeling.

The syntactic voting and semantic labeling parts of our system are implemented in Java together with a few Perl scripts. Using the open source TADM for parameter estimation, our the voting component take no more than 1 minute to train and 10 seconds to run (on WSJ testset). The SRL component takes about 1 hour for training, and no more than 30 seconds for labeling (WSJ testset).

Result analysis shows that the combination of the two state-of-the-art parsers delivers good syntactic dependencies (ranked 2nd). Error analysis shows most of the errors are related to prepositions. One category is the syntactic ambiguity of pp-attachment, e.g. in "when trading was halted in Philip Morris", "in" can be attached to either "trading" or "halted". The other category is the LOC and TMP tags in phrases like "at the end of the day", "at the point of departure", etc.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
	P lemma	P POS	P rel	P-parent POS	A lemma	A POS	A rel	P-children POSes	P-children rel	P-A path	A-children POSes	A-children rels	P precedes A?	A's position	P-siblings POSes	P-siblings rels	P NE label	P WN sense	P MRS EP-name	P MRS-args labels	P MRS-args POSes	A MRS EP-name	A MRS-preds labels	A MRS-preds POSes
PI	⊗	⊗	⊗	⊗				⊗	⊗						×	×			○	○	○			
AI	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗		×	×			○	○	○	○	○	○
AC	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	⊗	×	×			○	○	○	○	○	○
PC	⊗	⊗	⊗	⊗				⊗	⊗						×	×			○	○	○			

Table 1: Feature types used in semantic role labeling sub-components. Feature types marked with × are used in the "closed" run; feature types marked with ○ are used in the "open" run; feature types marked with ⊗ are used in both runs. P denotes predicate; A denotes semantic argument.

The results on semantic role labeling show, sometimes, even with syntactic errors of LOC/TMP tags, the semantic role labeler can still predict AM-LOC/AM-TMP correctly, which indicates the robustness of our hybrid approach. By comparing our "closed" and "open" runs, the MRS features do introduce a clear performance improvement. The performance gain is even more significant in out-domain test, showing that the MRS features from ERG are indeed much less domain dependent. Another example worth mentioning is that, in the sentence "Scotty regarded the ear and the grizzled hair around it with a moment of interest", it is extremely difficult to know that "Scotty" is a semantic role of "interest".

Also, we are the only group that submitted runs for both tracks, and achieved better performance in open competition. Although the best ways of integrating deep linguistic processing techniques remain as an open question, the achieved results at least show that hand-crafted grammars like ERG do provide heterogeneous linguistic insights that can potentially find their usage in data-driven NLP tasks as such.

6 Conclusion

In this paper, we described our hybrid system on both syntactic and semantic dependencies labeling. We built a voting model to combine the results of two state-of-the-art syntactic dependency parsers, and a pipeline model to combine deep parsing results for SRL. The experimental results showed the advantages of our hybrid strategy, especially on the cross-domain data set. Although the optimal ways of combining deep processing techniques remains to be explored, the performance gain achieved by incorporating hand-crafted grammar outputs shows a promising direction of study for both fields.

References

Callmeier, Ulrich. 2001. Efficient parsing with large-scale unification grammars. Master's thesis, Universität des Saarlandes, Saarbrücken, Germany.

Copestake, Ann, Dan Flickinger, Carl J. Pollard, and Ivan A. Sag. 2005. Minimal recursion semantics: an introduction. *Research on Language and Computation*, 3(4):281–332.

Flickinger, Dan. 2000. On building a more efficient grammar by exploiting types. *Natural Language Engineering*, 6(1):15–28.

Hacioglu, Kadri. 2004. Semantic role labeling using dependency trees. In *Proceedings of COLING 2004*, pages 1273–1276, Geneva, Switzerland, Aug 23–Aug 27.

McDonald, Ryan, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005. Non-Projective Dependency Parsing using Spanning Tree Algorithms. In *Proceedings of HLT-EMNLP 2005*, pages 523–530, Vancouver, Canada.

Nivre, Joakim, Jens Nilsson, Johan Hall, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(1):1–41.

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)*, Manchester, UK.

Zhang, Yi and Valia Kordoni. 2005. A statistical approach towards unknown word type prediction for deep grammars. In *Proceedings of the Australasian Language Technology Workshop 2005*, pages 24–31, Sydney, Australia.

Zhang, Yi, Valia Kordoni, and Erin Fitzgerald. 2007. Partial parse selection for robust deep processing. In *Proceedings of ACL 2007 Workshop on Deep Linguistic Processing*, pages 128–135, Prague, Czech.