Copenhagen-Malmö: Tree Approximations of Semantic Parsing Problems

Natalie Schluter†, Jakob Elming, Sigrid Klerke, Héctor Martínez Alonso, Dirk Hovy
Barbara Plank, Anders Johannsen, and Anders Søgaard

†Dpt. of Computer Science
Malmö University
natalie.schluter@mah.se

Center for Language Technology
University of Copenhagen
{zmk867,skl,alonso}@hum.ku.dk
{dirk,bplank}@cst.dk,
{ajohannsen,soegaard}@hum.ku.dk

Abstract

In this shared task paper for SemEval-2014 Task 8, we show that most semantic structures can be approximated by trees through a series of almost bijective graph transformations. We transform input graphs, apply off-the-shelf methods from syntactic parsing on the resulting trees, and retrieve output graphs. Using tree approximations, we obtain good results across three semantic formalisms, with a 15.9% error reduction over a state-of-the-art semantic role labeling system on development data. Our system came in 3/6 in the shared task closed track.

1 Introduction

Semantic analyses often go beyond tree-structured representations, assigning multiple semantic heads to nodes, some semantic formalisms even tolerating directed cycles.[1] At the same time, syntactic parsing is a mature field with efficient, highly optimised decoding and learning algorithms for tree-structured representations. We present tree approximation algorithms that in combination with a state-of-the-art syntactic parser achieve competitive performance in semantic digraph parsing.

We investigate two kinds of tree approximation algorithms that we will refer to as *pruning* algorithms and *packing* algorithms. Our pruning algorithms simply remove and reverse edges until the graph is a tree; edge reversals are then undone as a postprocessing step. Our packing algorithms, on the other hand, carry out two bijective graph transformations to pack structural information into new edge labels, making it possible to reconstruct most of the structural complexity as a postprocessing step. Specifically, we present a packing algorithm that consists of two fully bijective graph transformations, in addition to a further transformation that incurs only a small information loss.

We carry out experiments across three semantic annotations of the Wall Street Journal section of the Penn Treebank (Marcus et al., 1993), corresponding to simplified versions of the semantic formalisms minimal recursion semantics (MRS) (Copestake et al., 2005), Enju-style predicate-argument structures (Miyao and Tsujii, 2003), and Prague-style tectogrammar semantics (Böhmová et al., 2003). We show that pruning and packing algorithms lead to state-of-the-art performance across these semantic formalisms using an off-the-shelf syntactic dependency parser.

2 Related work

Sagae and Tsujii (2008) present a pruning algorithm in their paper on transition-based parsing of directed acyclic graphs (DAGs), which discards the edges of longest span entering nodes. They apply the dependency parser described in Sagae and Tsujii (2007) to the tree representations. We note that this algorithm is not sufficient to produce trees in our case, where the input graphs are not necessarily acyclic. It does correspond roughly to our LONGEST-EDGE baseline, which removes the longest edge in cycles, in addition to flow reversal.

Sagae and Tsujii (2008) also present a shift-reduce automaton approach to parsing DAGs. In their paper, they report a labeled F1-score of 88.7% on the PAS dataset (see Section 3), while we obtain 89.1%, however the results are thus not directly comparable due to different data splits.[2]

[1] For example, HPSG predicate-argument structures (Pollard and Sag, 1994).

[2] We obtained code to run this as a baseline, but were unable to, due to memory leaks, caused by subsets of our data, and on the subsets of data that actually parsed, recall was very

Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014), pages 213–217,
Dublin, Ireland, August 23-24, 2014.

The shared task organizers of the Broad-coverage Semantic Dependency Parsing task at SemEval-2014[3] also presented a pruning-based baseline system. They eliminate re-entrancies in the graph by removing dependencies to nodes with multiple incoming edges. Of these edges, they again keep the shortest. They incorporate all singleton nodes by attaching nodes to the immediately following node or to a virtual root - in case the singleton is sentence-final. Finally, they integrate fragments by subordinating remaining nodes with in-degree 0 to the root node. They apply the parser described in Bohnet (2010), also used below, to the resulting trees. This system obtained a labeled F1-score of 54.7% on the PAS dataset. The performance of their pruning algorithm was also considerably lower than our algorithms on the other datasets considered below.

3 Tree approximations

This section describes two approaches to approximating graphs by trees, namely pruning and packing. Pruning optimizes the number of "good" edges in trees (Section 3.1), while packing transforms graphs into trees by means of a pipeline of operations which are 99.6% reversible (see Figure 1); that is, almost no information from the original graphs is lost in the trees (Section 3.2).

Under both approaches, we first introduce artificial root nodes to the graphs and append them to the word list. Graphs may initially be disconnected. We connect all weakly connected components as follows. We first identify a most important node in each weakly connected component, which we will refer to as the *root*. This root is taken to be the first node with the "top" feature from the data, if one exists. If none exists, then the node with highest degree is chosen as the "root". (Note that the "root" of each non-singleton connected component is marked as a "top" node in the inverse transformation.) The root of each non-singleton weakly connected component is attached as a dependent of the artificial root node with a special new label for the corresponding edge. Also, each disconnected node is attached as a dependent of the node to the right of it, with a distinct special new label. It is these connected graphs that we take to be the input in the following two subsections describing our graph pruning and packing algorithms.

3.1 Graph pruning

Our PRUNING algorithm removes a small number of edges in the semantic graphs to be able to represent them as trees. The average edge counts from the training data (see Section 4.1) indicate that the potential edge loss in pruning is relatively small (5.7% in the worst case). In this approach, two transformations on the connected semantic graphs are carried out: pruning and flow reversal.

Pruning. The input digraph may contain underlying *undirected* cycles. We break these cycles by iteratively removing the longest edge from the node with the fewest predecessors (lowest depth) in the digraph. The resulting underlying undirected graph is a tree.

Depth-first flow reversal. We then carry out depth-first traversal of the resulting underlying undirected tree, reversing the direction of edges from the leaves upwards, as needed, until reaching the root. Any reversed edge's label is given a special prefix, so that this reversal can be undone in a post-processing step.

Following the above two transformations, we train our parsers on the transformed semantic annotations and output graphs such as the one in Figure 1a.

3.2 Graph packing

Our PACKING algorithm consists of a pipeline of four graph transformations. The two major transformations are for coordination and generalised long-distance dependencies, being both parallel path inducing constructions. The transformations are both linguistically and topologically inspired by the f-structure annotated c-structures in Lexical Functional Grammar and f-structure parsing via off-the-shelf dependency parsers (Schluter and Van Genabith, 2009). We further ensure the defining tree property that every node is connected by a unique path from the root, by carrying out flow reversal when necessary. Finally remaining parallel paths are broken according to an heuristic on path locality.

Coordination. In some semantic representations of coordination, individual conjunct nodes may all dominate a same argument, or be dominated by a same head. In both these cases, parallel paths are generated. The same structures may

low, suggesting that maybe the decoding algorithm was tuned to a specific planarization of the complex graphs.

[3] http://alt.qcri.org/semeval2014/task8/

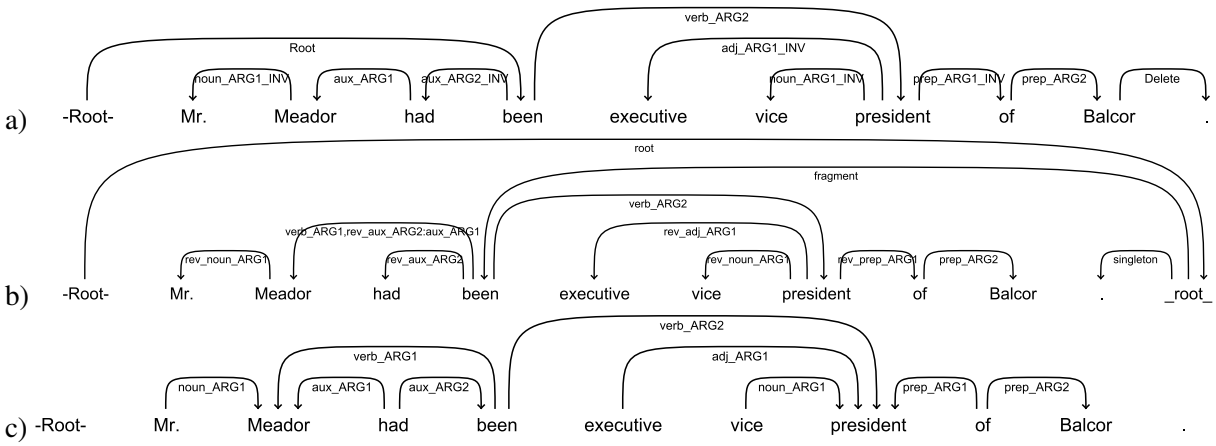

Figure 1: Example of pruned (top), packed (middle), and original (bottom) semantic graph. (Sentence 22002004 from the PAS dataset.)

be represented if the head or arguments are "factored out". To do this, we remove all edges from conjuncts towards a same argument (resp. from a shared head to each conjunct), and introduce a new edge from the root of the coordination subtree towards this argument (resp. from a shared head to the root of the coordination subtree). The new edges receive a special prefix to facilitate applying the inverse transformation.

Breadth-first flow reversal. Unlike our pruning algorithm, there is not yet any clear distinct path from the root to the all nodes (as there are not leaves yet). After carrying out the coordination transformation, we carry out a breadth-first search on the graph to direct flow away from the root, and again, reversed edges' labels are given a special prefix. As we do this, we test resulting nodes to see if there are any parallel paths leading to them. If so, these paths may be transformed immediately according to the following transformation.

Generalized long-distance dependencies. Long-distance dependencies are represented in f-structure annotated c-structures by path equations. This gives a tree representation of parallel paths, at least one of which is exactly one edge long. Given two parallel paths p_1 and p_2 in the graph, where $p_1 = (v_1, l, v_n)$ and $p_2 = (v_1, l_1, v_2), (v_2, l_2, v_3), \ldots, (v_{n-1}, l_{n-1}, v_n)$, we remove the last edge of p_2 and augment p_1's label with the representation $l_1 : l_2 : \cdots : l_{n-1}$ of p_2. p_1 becomes $(v_1, l \texttt{ and } l_1 : l_2 : \cdots : l_{n-1}, v_n)$, indicating that v_n is also the child (with dependency label l_{n-1}) of the node found by travelling (from v_1) down an l_1 labelled edge, followed by an l_2 labelled edge, and so on until the child of the l_{n-2} labelled edge is found.

Maximum average locality heuristic. Following these transformations, there may still be parallel paths in the graph: those not parallel to a single edge. We remove "worst" re-entrant edges using the simple heuristic that the path with the lowest average edge span should be conserved entirely. These removed edges clearly cannot be recovered after transformation.

Our parsers are trained on the output graphs of these four transformations such as the one in Figure 1b. We observe the main difference between PRUNING and PACKING: coordination and long-distance dependencies. For example, PACKING keeps the edge between the conjunction and the first conjunct, which is pruned away in PRUNING. Such a difference provides a partial explanation for the lower recall of PRUNING vis-à-vis PACKING (see Section 4.5).

4 Experiments

4.1 Data

The three datasets are semantic annotations of the WSJ section of the Penn Treebank of English. The average sentence length, which is also the average number of dependency edges in the tree approximations that we use to induce our semantic parsers, is 22.93. The three semantic formalisms are slightly richer, and the average number of edges in the PAS-annotated treebank is 24.32. For DM, the average number of edges is 23.77, and for DM it is 23.33. While the pruning-based approaches thus suffers from a modest information loss, throwing out 5.7% of the edges in the worst

case, this is not the case for packing. The reversibility of the packed representations is given by the score upper bound in the last row in Table 1. We use the dataset splits of the SemEval 2014 shared task.

4.2 Model

For both our pruning and packing models, we use the Mate parser (Bohnet, 2010)[4] with default parameters to learn our parsing models. The Mate parser is trained on the output of the transformation pipeline on Sections 00-19 of the three semantically annotated WSJ datasets. Some models use Brown clusters generated from Sections 00-19 only. This does not solve OOV problems, but allows of slightly better generalisation across distributionally similar words in the training data.

4.3 Baselines

We use the SemEval 2014 shared task baseline (SIMPLE-PRUNE; see Section 2), as well as the LONGEST-EDGE baseline, also mentioned above. The latter is our strongest baseline system. It is very similar to PRUNING, in doing both edge pruning and flow reversal, but the pruning step only removes the longest edge rather than considering node depth. Our third baseline is the Mate semantic role labeling learner (SRL-DEP) (Björkelund et al., 2009), which uses predicted syntactic parses as input; for this, we use the syntactic parses made available in the SemEval 2014 shared task for replicability.

Approach	**Cl**	DM	PAS	PCEDT	*Av*
Systems					
PRUNING	NO	86.6	88.8	72.7	82.7
	YES	**86.9**	**89.1**	72.5	**82.8**
PACKING	NO	85.8	88.7	71.8	82.1
	YES	86.1	88.7	**72.9**	82.6
Baselines					
SIMPLE-PRUNE		54.7	50.9	67.8	57.8
LONGEST-EDGE		83.8	88.9	66.1	79.6
SRL-DEP		79.5	82.4	70.1	77.4
Upper bound					
PACKING		*99.9*	*99.5*	*99.5*	*99.6*

Table 1: Labelled F1-score results on development data, with and without use of Brown clusters (**Cl**).

4.4 Results

The results are presented in Tables 1 through 3, where the system evaluations for the SemEval task are marked with asterisks in Table 2. We note that all our approaches do considerably better than our three baselines. The error reduction of our best system over the SRL system across all three formalisms is 24.2%, and the error reduction over the more competitive pruning baseline LONGEST-EDGE is 15.9%. As mentioned in Section 2, these results seem to promise better performance than current DAG parsing models. Note from the results in Table 2 that, as expected, PRUNING leads to higher precision than PACKING at the expense of recall.

[4]https://code.google.com/p/mate-tools/

Approach	metric	DM	PAS	PCEDT	*Av*
Systems					
PACKING	PREC	84.8	87.7	71.2	81.2
(W/ TOP)	REC	84.0	88.4	68.6	80.3
	F1	**84.4**	**88.0**	**69.9**	**80.8***
	PREC	85.4	87.9	70.8	81.4
(W/O TOP)	REC	84.6	88.6	68.8	80.7
	F1	**85.0**	**88.3**	**69.9**	**81.1**
PRUNING	PREC	87.2	91.3	72.8	83.8
(W/ TOP)	REC	80.2	81.3	62.8	74.8
	F1	**83.6**	**86.0**	**67.4**	**79.0***
	PREC	87.2	91.3	72.8	83.8
(W/O TOP)	REC	85.1	85.1	68.0	79.4
	F1	**86.2**	**88.1**	**70.3**	**81.5**

Table 2: Labelled results on test data, with and without evaluation of top nodes. The scores with asterisks correspond to the output evaluated in the SemEval task.

Approach	metric	DM	PAS	PCEDT	*Av*
Systems					
PACKING	PREC	86.8	89.1	84.8	86.9
(W/ TOP)	REC	86.0	89.8	81.8	85.9
	F1	**86.4**	**89.4**	**83.2**	**86.3**
	PREC	87.5	89.4	85.4	87.4
(W/O TOP)	REC	86.7	90.1	83.0	86.6
	F1	**87.1**	**89.7**	**84.2**	**87.0**
PRUNING	PREC	89.2	92.6	88.2	90.0
(W/ TOP)	REC	82.0	82.5	76.1	80.2
	F1	**85.4**	**87.3**	**81.7**	**84.8**
	PREC	89.2	92.6	88.2	90.0
(W/O TOP)	REC	87.1	86.3	82.4	85.3
	F1	**88.1**	**89.3**	**85.2**	**87.5**

Table 3: Unlabelled results on test data, with and without evaluation of top nodes.

4.5 Error Analysis

We observe that pruning leads to high precision, while our packing algorithm gives us much better recall. This is not surprising, since our packed representations introduce new labels, making it harder to generalize at training time. On the other hand, pruning approaches suffer in recall, simply because edges are thrown away in preprocessing the data.

5 Conclusions

In this paper, we experimented with using tree approximation algorithms to reduce semantic structures to trees and use off-the-shelf structured prediction techniques to train semantic parsers. Our approximation algorithms include both pruning and packing algorithms, i.e., algorithms that try to reduce graphs to trees optimally, as well as algorithms that pack information about graphs into trees from which we later recover the richer structures. Using these tree approximation algorithms, we obtain 15.9% error reductions over a state-of-the-art SRL system.

References

Anders Björkelund, Love Hafdell, and Pierre Nugues. 2009. Multilingual semantic role labeling. In *Proc. of CoNLL: Shared Task*, pages 43–48, Boulder, CO, USA.

Alena Böhmová, Jan Hajič, Eva Hajičová, and Barbora Hladká. 2003. The Prague Dependency Treebank: A three-level annotation scenario. In Anne Abeillé, editor, *Treebanks: Building and Using Syntactically Annotated Corpora*, pages 103–127. Kluwer, Netherlands.

Bernd Bohnet. 2010. Top accuracy and fast dependency parsing is not a contradiction. In *Proc. of COLING*, pages 89–97, Beijing, China.

Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan Sag. 2005. Minimal recursion semantics. *Research on Language and Computation*, 3:281–332.

Mitchell Marcus, Mary Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Yusuke Miyao and Jun'ichi Tsujii. 2003. Probabilistic modeling of argument structures including non-local dependencies. In *Proc. of RANLP*, pages 79–85, Borovets, Bulgaria.

Carl Pollard and Ivan Sag. 1994. *Head-driven phrase structure grammar*. University of Chicago Press.

Kenji Sagae and Jun'ichi Tsujii. 2007. Dependency parsing and domain adaptation with LR models and parser ensembles. In *Proc. of CoNLL Shared task session of EMNLP-CoNLL*, pages 1044–1050, Prague, Czech Republic.

Kenji Sagae and Jun'ichi Tsujii. 2008. Shift-reduce dependency DAG parsing. In *Proc. of COLING*, pages 753–760, Manchester, UK.

Natalie Schluter and Josef Van Genabith. 2009. Dependency parsing resources for French. In *Proc. of NODALIDA*, pages 166–173, Odense, Denmark.