Maximising spanning subtree scores for parsing tree approximations of semantic dependency digraphs

Natalie Schluter
Center for Language Technology
University of Copenhagen
natschluter@hum.ku.dk

Abstract

We present a method for finding the best tree approximation parse of a dependency digraph for a given sentence, with respect to a dataset of semantic digraphs as a computationally efficient and accurate alternative to DAG parsing. We present a training algorithm that learns the spanning subtree parses with the highest scores with respect to the data, and consider the output of this algorithm a description of the best tree approximations for digraphs of sentences from similar data. With the results from this approach, we acquire some important insights on the limits of solely data-driven tree approximation approaches to semantic dependency DAG parsing, and their rule-based, pre-processed tree approximation counterparts.

1 Introduction

In semantic dependency parsing, the aim is to recover sentence-internal predicate argument relationships; structurally speaking, given a sentence, the objective is to recover the possibly disconnected digraph (which represents the semantic structure of the sentence). The sparsity of digraph representations of some semantic dependency digraph datasets (i.e., the fact that the number of edges is linear in the number of vertices), as well as the well-performing first attempts at such tree approximations from Schluter et al. (2014), Agić and Koller (2014), and Agić et al. (2015) suggest that tree approximations for digraph semantic dependency structures are a relevant avenue to the development if not the sidestepping of some computationally harder models of directed acyclic graph (DAG) and digraph decoding (McDonald and Pereira, 2006; Martins and Almeida, 2014).

In this paper, we present a simple adaptation of the passive-aggressive perceptron training algorithm used in (Crammer et al., 2006; Björkelund et al., 2009) to the task of finding the parameters that describe the highest scoring tree approximations of semantic dependency digraphs, given a training corpus. The key change in the algorithm is to iteratively minimise the error in precision between output spanning subtrees and corresponding training digraph instances, allowing therefore the algorithm to choose best spanning subtree approximations with respect to the dataset, rather than forming tree approximations as a pre-processing step to training as was done by Schluter et al. (2014), Agić and Koller (2014), and Agić et al. (2015).

Because we directly adapt the software used by (Björkelund et al., 2009), without increasing any computational complexity, the approach also benefits from a syntactic parsing algorithm optimised both practically and theoretically for efficiency, robustness, and accuracy. Supposing these natural tree patterns exist in the data, the approach intuitively is very attractive, since it lets natural patterns of the data dictate tree approximations, rather than depending on either the previous knowledge of parser behaviour or anecdotal linguistic knowledge. Moreover, the approach promises some insight into the nature of the patterns inherent in the semantic digraph data, reflecting, namely, the question of the existence of unique most likely sub-tree structures in the data and why rule-based pre-processing tree approximations work so well for semantic digraph dependency parsing.

2 Previous work and motivation

In the context of the SemEval task 8 on Broad Coverage Semantic Dependency Parsing, Schluter et al. (2014) and Agić and Koller (2014) introduce a tree approximation definition for the semantic dependency parsing task, including a number of different approaches. Both present similar pruning pre-processing steps as a potential approach. Additionally, Schluter et al. (2014) present a packing pre-processing step as a further approach. We discuss the approaches presented in (Schluter et al., 2014), because we adopt the parts of their pre-processing which do not remove any edges. Their approaches also outperformed that of (Agić and Koller, 2014) in the SemEval task.

In (Schluter et al., 2014), all digraphs were first transformed into rooted DAGs (that is, with a single root, from which all nodes are reachable), following which the authors experimented with creating the tree approximations for each of the training set digraphs individually as follows.

In their first approach, (Schluter et al., 2014) consider underlying undirected cycles of digraphs and **pruned** these digraphs by removing from the cycle the longest edge from the node with the fewest predecessors (lowest depth) in the digraph. In their second approach, they attempt to store almost all digraph information into the graph, by **packing** parallel path information, structurally corresponding to long-distance dependency re-entrancies, into a single complex edge label; these removed edges can then recovered as a post-processing step, by "expanding" the complex edge label. However, not all parallel path sets between two nodes include a path of length 1 edge; for these, an heuristic approach was taken whereby only the edge of shortest span from among all of the last edges of these paths was retained.

The result from both these approaches were trees that were used as tree approximations to training set digraphs, individually. The packing approach suffered from complex edge labels with low frequency in the training set, and which could not always be resolved fully in post-processing at test-time; as a result, this approach had relatively stable precision and recall (with respect to each other), but with lower precision than the pruning approach, which achieved high precision, but low recall. Upon carrying out this research, we posited that the lower recall of the pruning approach was caused by not accounting for the tree approximations with respect to the entire dataset, rather only with respect to a structural heuristic on individual digraphs of the training set.

The motivation for this work is therefore an attempt to attain similar precision to the pruning approach in (Schluter et al., 2014), with similar recall to the same authors' packing approach, by only pruning edges, but leaving it to the parser to determine, given the training data, which edges should be pruned. The sparsity of the graphs in all three datasets (Cf. Section 4) as well as the disparity in precision and recall between the approaches in (Schluter et al., 2014) (Cf. Section 5) suggests that there is room for improvement in tree approximation approaches to semantic dependency parsing for these particular datasets.

Further improvements to the pre-processing approach to tree approximation parsing of semantic digraphs has been obtained by Agić et al. (2015). This approach was guided by both the ideas of pruning and packing: they first present a study on the types of re-entrancies displayed by the digraph data, to discover that many of them are predictable. Such re-entrancy information needs not be packed into other labels as it is done by (Schluter et al., 2014), as this pruning is nearly 100% reversible. These observations provide further insight into the performance of the approach presented here, given its behaviour as well as the dataset used. Indeed, persistent non-tree-like structures in the training data will inhibit training algorithms from aggressively deciding on a best tree-like substructure (Cf. Section 5).

The (non-ensemble approach) state of the art on the datasets we experiment on is achieved using a second-order model with approximate digraph decoding via alternate directions dual decomposition (Martins and Almeida, 2014; Martins et al., 2011).

3 Finding the best maximum spanning subtree approximation

As an approximate approach to finding the best scoring semantic dependency parse digraph given a training corpus of such digraphs, we present an approach to finding the best scoring semantic dependency spanning subtree parse, given a training corpus of semantic dependency digraphs. That is, we adopt a second-order approach to the problem of finding the best semantic dependency parse digraph. Our original objective was to find

$$D^* = \arg\max_{D \in \mathcal{D}(x)} \phi_D(\mathbf{w}, x, D),$$

where $\mathbf{w}$ is a weight vector, D is a dependency digraph and x is the input sentence. The best dependency digraph is the output according to a scoring function ϕ_D—a sum over sub-factors representing some second-order description of the dependency structure. The parameter $\mathbf{w}$ is obtained by some algorithm that minimises the error defined by a distance function between digraph parse D'_i and training digraph D_i.

Our aim here is to approximate D^* by a tree T^*, so our objective now concerns possible trees $T \in \mathcal{T}$; we want to find the tree that maximises the scoring function ϕ_T for trees,

$$T^* = \arg\max_{T \in \mathcal{T}(x)} \phi_T(\mathbf{w}, x, T),$$

where ϕ_T is composed of second-order factors describing (and, practically speaking, coincides precisely with the scoring function ϕ from (Carreras, 2007)). However, we obtain the parameter $\mathbf{w}$ by training on a dataset of dependency digraphs and carefully minimising the error between these two very different types of structures, since a tree can never contain more than $|x| - 1$ edges and there is therefore a risk of non-convergence if the wrong error measure is chosen.

We directly adapt the mate parser (Bohnet, 2010) for this modified task. To do this, we simply adjust the unregularised passive-aggressive perceptron algorithm implementation used in training by the parser, the averaged-perceptron version of which was presented first in (Carreras, 2007), in three key ways. First, edge features are taken from entire digraphs, rather than just trees. These features, unlike in (Martins and Almeida, 2014), do not account for multiple heads in digraphs, because we need to use the scoring function on trees rather than digraphs. Secondly, the error is minimised between the original graphs and the maximum spanning subtree implied by $\mathbf{w}$. In doing so, the algorithm finds weight vectors that minimise the error of the tree approximation of the graph. Finally, we changed the error function from $(1-\text{recall})$ in the original version of the mate parser to $(1-\text{precision})$ to avoid punishing trees for not being digraphs in the updates, and thereby prevent non-convergence.

Algorithm 1 shows the adaptation, where an instance is a pair (x_j, D_j) of x_j, a sentence, and D_j, a dependency digraph. The algorithm begins by initialising the weight vector $\mathbf{w}$ to the zero vector. It then extracts the features from the training set, $Z : \{(x_j, D_j)\}_{j=1}^{N}$ and stores them on the hard disk, after which training is carried out using the passive-aggressive algorithm. Iteratively, it (1) reads in the features and calculates possible edge weights, (2) decodes using the second order extension of the Eisner algorithm for projective dependency trees (Eisner, 1996) presented in (Carreras, 2007), and then (3) tries to find a higher scoring non-projective tree by exchanging edges out of the output from (2) using the edge weights in (1). Details for the original algorithm can be found (Bohnet, 2010); however, note that in Algorithm 1, $\hat{T}$ is a dependency tree and D_j is a dependency digraph. Also, error is measured in terms of precision (for Lines 13 and 14).

Algorithm 1 **`Training`** (Z)
// where $Z = \{(x_j, y_j)\}_{j=1}^{N}$ is the digraph training data

```
1: $\mathbf{w} \leftarrow \mathbf{0}$
2: for $j \leftarrow 1$ to $N$ do
3:     extract-and-store-features($x_j$)
4: end for
5: for $i \leftarrow 1$ to $I$ do
6:     // where $I$ is the number of iterations
7:     for $j \leftarrow 1$ to $N$ do
8:         $k \leftarrow (i-1) * N + j$
9:         $\gamma \leftarrow I \times N - k + 2$   // passive-aggressive weight
10:        $A \leftarrow$ read-features-and-calc-arrays($j, \mathbf{w}$)
11:        $\hat{T} \leftarrow$ predict-projective-parse-tree($A$)
12:        $\hat{T} \leftarrow$ non-projective-approx($\hat{T}, A$)
13:        $e \leftarrow \Delta(\hat{T}, D_j)$   // the error
14:        $\mathbf{w} \leftarrow$ update($e, \gamma$)
15:    end for
16: end for
```

4 The graphs and their tree-likeness

Organisers for the SemEval task 8 on Broad Coverage Semantic Dependency Parsing (SDP) (Oepen et al., 2014) proposed three different annotations for evaluation of the task, resulting in the three semantically annotated datasets, over the same text—that is, the Wall Street Journal section of the English Penn Treebank. As in the original task, we refer to the datasets as

DM: a transformation of Flickinger et al. (2012)'s DeepBank by Miyao et al. (2014)'s system.

PAS: the predicate-argument structures of the WSJ portion of the HPSG treebank.

PCEDT: the tectogrammatical layer of annotations from the Prague Czech-English Dependency Bank (Hajič et al., 2012).

Graph sparsity and treewidth. We note that the average number of edges of a spanning subtree for training set graphs is 22.93. On the other hand, after the aforementioned pre-processing, the average number of edges in the DM-annotated treebank is 23.77, PAS-annotated treebank is 24.32, and for PCEDT it is 23.33. By removing edges from digraphs to make spanning subtrees, we thus lose at most 5.7% edges, showing the general sparsity of digraphs in these datasets and further showing the relevance of the tree approximation approach. Agić et al. (2015) calculate the average treewidth of the underlying undirected graphs of SDP digraphs, to be 1.3 for DM, 1.71 for PAS, and 1.45, for PCEDT, indicating that DAG parsing algorithms heavily based on the more efficient tree parsing algorithms are a promising avenue for further research.

4.1 Pre-processing the graph data

In general, the digraphs amongst three datasets are disconnected. In order for the solution of finding a most likely tree approximation to make any sense, therefore, we transformed digraphs similarly to the approach taken in (Schluter et al., 2014): (1) a dummy root node is placed to the left of the input, (2) the top node is connected (as a child) to the dummy root node, (3) the node of highest degree (=indegree + outdegree) for non-singleton remaining weakly connected components is attached as a child of the dummy root node, and (4) all singleton weakly connected components are connected a child of the node to the left.

In most digraphs of of the three datasets, there is not any existing spanning subtree. Therefore, we carry out flow reversal for rooted DAG construction as a further pre-processing step. As in (Schluter et al., 2014), we created rooted DAGs during a breadth-first search of the digraph, reversing the direction of edges when necessary, and marking the label of reversed edges (for reversibility of the transformation). So, our label sets for the three datasets may at most double, which admittedly increases (by a factor of 2) instance size and therefore running time.

5 Experiments and error analysis

For our experiments, we used precisely the same data split as in SemEval 2014's task 8 and the original mate parser default parameters for the modified version, with the exception that we increased the number of iterations to 15. The results are given in Table 1. Compared with the preprocessing tree approximations from (Schluter et al., 2014), the subgraph score maximisation approach performs quite poorly. The approach successfully closes the gap between precision and recall, when compared to the pruning approach in (Schluter et al., 2014), but both precision and recall are relatively low.

	data	LP	LR	LF	UP	UR	UF
pack	DM	84.8	84.0	84.4	86.8	86.0	86.4
	PAS	87.7	88.4	88.0	89.1	89.8	89.4
	PCEDT	71.2	68.6	69.9	84.8	81.8	83.2
prune	DM	87.2	80.2	83.6	89.2	82.0	85.4
	PAS	91.3	81.3	86.0	92.6	82.5	87.3
	PCEDT	72.8	62.8	67.4	88.2	76.1	81.7
this paper	DM	67.2	69.7	68.4	69.8	72.4	71.1
	PAS	83.1	77.5	80.2	86.4	80.7	83.6
	PCEDT	62.3	58.5	60.4	79.0	74.3	76.6

Table 1: Precision, recall and f-score over the three datasets (pack and prune results are from (Schluter et al., 2014)) .

An analysis of the pre-processing (Schluter et al., 2014; Agić and Koller, 2014; Agić et al., 2015) versus statistical (this paper) approaches may provide some insight as to why. Especially Agić et al. (2015) show that many non-tree-like structures of the SDP data are predictable; that is, we can turn them into trees by consistently pruning edges and "understand" that they are in fact more complex than trees, without encoding this information into the resulting pruned tree in any way. In a post-processing step, these edges are simply reintroduced using some rules. These are edges for which it would be difficult for the passive aggressive algorithm that we employ to choose between, since there are virtually no structures that require only a subset of them. As a result, making a strict rule about what tree structures should be predicted as a pre-processing step results in better tree approximations than the purely data-driven approach presented here.

6 Concluding remarks

We have presented an approach to semantic dependency parsing that takes advantage of and subtly adapts an efficient and highly optimised syntactic dependency tree parsing system to the job of finding best tree approximations of digraphs. Intuitively, the approach is attractive, because it requires the data to choose digraph approximations, rather than using any anecdotal linguistic knowledge to hard-code a pre-processor to build the approximations before training. However, the approach fails to outperform rule-based preprocessing for tree approximations, seemingly because there is often no clear statistical preference among various subtrees of DAGs in the SDP data. The performance of a combination of the data-driven and rule-based preprocessing methods remains a viable open question.

References

Željko Agić and Alexander Koller. 2014. Potsdam: Semantic dependency parsing by bidirectional graph-tree transformations and syntactic parsing. In *Proc of Semeval*, Dublin, Ireland.

Željko Agić, Alexander Koller, and Stephan Oepen. 2015. Semantic dependency graph parsing using tree approximations. In *Proc of IWCS*, London, UK.

Anders Björkelund, Love Hafdell, and Pierre Nugues. 2009. Multilingual semantic role labeling. In *CoNLL*.

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

Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proc. of CoNLL Shared Task Session of EMNLP-CoNLL*, pages 957–961, Prague, Czech Republic.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *JMLR*, 7:551–585.

Jason Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of COLING-96*, pages 340–345, Copenhagen, Denmark.

Daniel Flickinger, Yi Zhang, and Valia Kordoni. 2012. Deep-bank. a dynamically annotated treebank of the wall street journal. In *Proc. of TLT*, pages 85–96, Lisbon, Portugal.

Jan Hajič, Eva Hajičová, Jarmila Panevová, Petr Sgall, Ondřej Bojar, Silvie Cinková, ..., and Zdeněk Žaborkrtský. 2012. Announcing prague czech-english dependency treebank 2.0. In *Proc. of LREC*, pages 3153–3160, Istanbul, Turkey.

André F. T. Martins and Mariana S. C. Almeida. 2014. Priberam: A turbo semantic parser with second order features. In *Proc of SemEval*, Dublin, Ireland.

André Martins, Noah A. Smith, M. A. T. Figueiredo, and P. M. Q. Aguiar. 2011. Dual decomposition with many overlapping components. In *Proceedings of EMNLP*.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proc. of EACL*, pages 81–88.

Yusuke Miyao, Stephan Oepen, and Daniel Zeman. 2014. In-house: An ensemble of pre-existing off-the-shelf parsers. In *Proc. of SemEval*.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajič, Angelina Ivanova, and Yi Zhang. 2014. Semeval 2014 task 8: Broad-coverage semantic dependency parsing. In *Proc. of SemEval*, pages 63–71, Dublin, Ireland.

Natalie Schluter, Anders Søgaard, Jakob Elming, Dirk Hovy, Barbara Plank, Hector Martinez Alonso, Anders Johanssen, and Sigrid Klerke. 2014. Copenhagen: Tree approximations of semantic parsing problems. In *Proceedings of SemEval*, Dublin, Ireland.