University of Toronto, Department of Computer Science
CSC 2501F—Computational Linguistics, Fall 2024

Reading assignment 2

Due date: Electronically by 12pm, Monday 30 September 2024.
Late write-ups will not be accepted without documentation of a medical or other emergency.
This assignment is worth 3% of your final grade.

What to read

Kurtz, R. et al. (2019) Improving Semantic Dependency Parsing with Syntactic Features. In *Proceedings of the First NLPL Workshop on Deep Learning for Natural Language Processing, 22nd Nordic Conference on Computational Linguistics*, Linköping University Electronic Press, pp. 12–21.

What to write

Write a *brief* summary of the paper's argumentation, with a critical assessment of its merits. Indicate how the material relates to the discussion of parsing that we have had in class.

Some points to consider:

- What is "semantic parsing?"
- Do we need syntax in semantic parsing?

General requirements: Your write-up should be typed, using 12-point font and 1.5-line spacing; it should fit on one to two sides of a sheet of paper. Submit using the `teach.cs submit` command.

Submission is electronic, through Quercus.

Improving Semantic Dependency Parsing with Syntactic Features

Robin Kurtz, Daniel Roxbo, Marco Kuhlmann

Linköping University
Department of Computer and Information Science
robin.kurtz@liu.se, marco.kuhlmann@liu.se

Abstract

We extend a state-of-the-art deep neural architecture for semantic dependency parsing with features defined over syntactic dependency trees. Our empirical results show that only gold-standard syntactic information leads to consistent improvements in semantic parsing accuracy, and that the magnitude of these improvements varies with the specific combination of the syntactic and the semantic representation used. In contrast, automatically predicted syntax does not seem to help semantic parsing. Our error analysis suggests that there is a significant overlap between syntactic and semantic representations.

1 Introduction

Semantic dependency parsing (SDP) is the task of mapping a sentence into a formal representation of its meaning in the form of a directed graph with arcs between pairs of words. Ever since the release of the now-standard datasets for this task (Oepen et al., 2014, 2015), most of the approaches to semantic dependency parsing have been based on previous and ongoing work in syntactic parsing. In particular, several semantic parsers make use of features defined over syntactic dependency trees; one recent example is the system of Peng et al. (2018).

In this paper we study to what extent semantic dependency parsing actually benefits from syntactic features. More specifically, we carry out experiments to identify those combinations of semantic and syntactic representations that yield the highest parsing accuracy. This is interesting not only for parser developers – using improvement or non-improvement in parsing accuracy as an indicator, our study also contributes to a better understanding of the similarities and contentful differences between semantic and syntactic representations.

Semantic dependency parsers are typically conceptualized as systems for structured prediction, combining a data-driven component that learns how to score dependency graphs with a decoder that retrieves one or several highest-scoring target graphs from the exponentially large search space of candidate graphs. Among decoding algorithms we find approaches based on integer linear programming (Almeida and Martins, 2015; Peng et al., 2017), dynamic programming algorithms that support exact decoding for restricted classes of graphs (Kuhlmann and Jonsson, 2015; Cao et al., 2017), and transition-based approaches introducing new shift–reduce-style automata (Zhang et al., 2016; Wang et al., 2018). Regarding the learning component, state-of-the-art parsing results have been achieved using neural architectures (Peng et al., 2017; Wang et al., 2018; Dozat and Manning, 2018). The system of Dozat and Manning (2018) even draws essentially all of its strength from its neural core, employing a trivial decoder. The parser used in this paper is a (slightly modified) re-implementation of that system developed by Roxbo (2019), which adds syntactic information via a simple head feature, along the lines of Peng et al. (2018).

Paper Structure. After providing some background in Section 2, we describe the architecture of our parser in Section 3, and our data and experimental setup in Section 4. In Section 5 we present our empirical results and complement them with an error analysis in Section 6. Section 7 concludes the paper and provides an outlook on future work.

2 Background

In both semantic and syntactic dependency parsing, the target structures are directed graphs with lexicalized nodes and bilexical arcs. More formally, a *dependency graph* for a sentence $\boldsymbol{x} = x_1, \ldots, x_n$ is an arc-labelled directed graph whose nodes are in

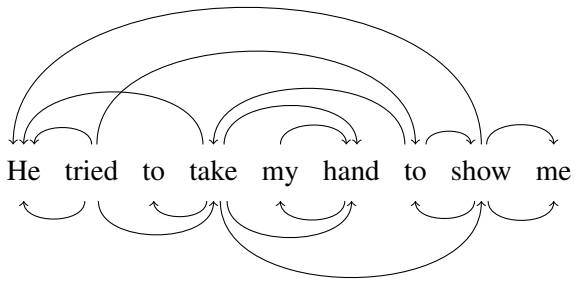


Figure 1: A sample sentence with a gold-standard semantic dependency graph in the DM representation (Flickinger et al., 2016, #41526060) (upper half-plane) and a predicted syntactic dependency tree in the Stanford Basic representation (lower half-plane). (Arc labels omitted in this example.)

one-to-one correspondence to the tokens of $\boldsymbol{x}$. For an arc $i \rightarrow j$ we refer to the nodes i and j as *head* and *dependent*, respectively. We follow standard conventions and visualize dependency graphs with their nodes laid out on a line according to the linear order of the sentence, and their arcs drawn in the half-plane above (or sometimes below) the nodes. An example graph is provided in Figure 1.

In syntactic dependency parsing, target representations are restricted to trees. Formally, a *dependency tree* is a dependency graph that is connected, acyclic, and such that each node except a distinguished root node has at most one incoming arc. The root node has no incoming arc. In a dependency tree we write $h(i)$ to denote the head of the incoming arc to the (non-root) node i.

3 Parser

We now give a compact description of our parser, a version of the system of Dozat and Manning (2018); for more details, we refer to Roxbo (2019). We use essentially the same architecture for semantic parsing and for predicting the trees over which we define our syntactic features.

3.1 Neural Network Model

The core of our parser is a bidirectional recurrent neural network with Long Short-Term Memory cells (BiLSTM; Hochreiter and Schmidhuber, 1997). Feeding this network with a sentence $\boldsymbol{x} = x_1, \ldots, x_n$ in the form of a sequence of (initially random) word embeddings $\boldsymbol{w}_i$, we obtain a sequence of context-dependent embeddings $\mathbf{c}_i$:

$$\boldsymbol{c}_1, \ldots, \boldsymbol{c}_n = \text{BiLSTM}(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_n)$$

The word embeddings can be easily augmented by additional lexical features, such as pre-trained word embeddings or embeddings for part-of-speech tags or lemmas. In this study we add embeddings created via character-based LSTMs and 100-dimensional GloVe (Pennington et al., 2014) embeddings.

The network processes the contextual token embeddings $\boldsymbol{c}_i$ by two parallel feedforward neural networks (FNN), which are meant to learn specialized representations for the potential roles of each word as head and dependent:

$$\boldsymbol{h}_i = \text{FNN}_h(\boldsymbol{c}_i) \qquad \boldsymbol{d}_i = \text{FNN}_d(\boldsymbol{c}_i)$$

These embeddings are then used to score each potential arc $i \rightarrow j$ via a bilinear model with weight matrix $\boldsymbol{U}$ that will be learned during training:

$$\text{score}(\boldsymbol{h}_i, \boldsymbol{d}_j) = \boldsymbol{h}_i^\top \boldsymbol{U} \boldsymbol{d}_j$$

3.2 Decoding

The matrix of arc scores can be processed by any type of arc-factored decoder to return the highest-scoring graph for the complete sentence. Our semantic parser greedily selects all arcs with a non-negative score. The syntactic parser uses the Chu–Liu/Edmonds (CLE) maximum spanning tree algorithm (Chu and Liu, 1965; Edmonds, 1967) implemented in Uniparse (Varab and Schluter, 2018).

3.3 Adding Labels

To predict labelled arcs, we take two different approaches: The semantic parser computes, for each token pair, scores for all potential labels, including a special NONE label that represents the absence of an arc between the two tokens; this yields a three-dimensional score tensor rather than a score matrix. The syntactic parser factorizes the computation and predicts a label for each token pair independently of the arc scorer; this label is only used if an arc is actually selected by the decoder.

3.4 Adding Syntactic Features

To add syntactic features to our semantic parser, we follow the same simple approach as Peng et al. (2018): Before feeding the contextual embedding of each token to the arc- and label-scoring components, we extend it with the contextual embedding of its syntactic head in the dependency tree:

$$\boldsymbol{c}'_i = [\boldsymbol{c}_i; \boldsymbol{c}_{h(i)}]$$

This simple variation has a minimal impact on the complexity of the overall model, and makes further analysis and comparison more straightforward.

Parameter	Value
Embeddings	100
Char LSTM	1 @ 400
Char linear	100
BiLSTM	3 @ 600
Arc/Label FNN	600
Epochs	100
Mini-batch size	50
Adam β_1	0
Adam β_2	0.95
Learning rate	$1 \cdot 10^{-3}$
Gradient clipping	5
Interpolation constant	0.025
L_2 regularization	$3 \cdot 10^{-9}$

Table 1: Network sizes and training parameters.

3.5 Training

We train our parser with the Adam optimizer (Kingma and Ba, 2014) and mini-batching. To train the arc and label scorers, we use a binary and a softmax cross-entropy loss, respectively. In the factorized approach used by the syntactic parser, the arc- and label-specific losses are summed up to an overall loss, weighted by an interpolation constant to emphasize the arc scorer:

$$\text{loss}_{\text{total}} = (1 - 0.025) \cdot \text{loss}_{\text{arc}} + 0.025 \cdot \text{loss}_{\text{label}}$$

Due to the size of the model and its fairly large number of trainable parameters (see Table 1), it is prone to overfitting. To address this, we apply equally large dropout rates to nearly all parts of the model (see Table 2). We apply variational dropout (Gal and Ghahramani, 2016) sharing the same dropout mask between all time steps in a sequence. On the LSTM hidden states we use DropConnect (Wan et al., 2013; Merity et al., 2017), a more general variant of dropout which drops individual connections instead of complete nodes of the computation graph.

Substructure	Rate
Embeddings	20%
Char LSTM feedforward	33%
Char LSTM recurrent	33%
Char Linear	33%
BiLSTM feedforward	45%
BiLSTM recurrent	25%
Arc FNN	25%
Arc scorer	25%
Label FNN	33%
Label scorer	33%

Table 2: Dropout rates.

4 Method

In this section we describe our data and the setup of our experiments.

4.1 Data

The main dataset for our experiments is the English part of the standard SDP distribution (Flickinger et al., 2016), which contains semantic dependency graphs for Sections 00–21 of the venerable Penn Treebank (Marcus et al., 1993) in a predefined train/test split, as well as graphs for out-of-domain test sentences from the Brown Corpus (Francis and Kučera, 1985). The graphs come in four different representation types, of which we use three: graphs derived from DeepBank (DM, Oepen and Lønning, 2006; Ivanova et al., 2012); predicate–argument structures computed by the Enju parser (PAS, Miyao, 2006); and graphs derived from the tectogrammatical layer of the Prague Dependency Treebank (PSD, Hajic et al., 2012). Due to their structural differences, the three graph types are more or less difficult to parse into; PSD graphs, for example, feature a considerably larger label inventory than the other types.

The graphs in the SDP dataset come with different types of gold-standard syntactic analyses, of which we use Stanford Basic Dependencies (SB, de Marneffe and Manning, 2008), derived from the Penn Treebank, and DELPH-IN Syntactic Derivation Trees (DT, Ivanova et al., 2012), derived from DeepBank. In addition to those we also use the English Web Treebank (EWT) from the Universal Dependencies (UD) project (Nivre et al., 2017), which contains syntactic dependency trees for text that does not overlap with the SDP data. We note that the EWT is considerably smaller than the SDP dataset.

4.2 Experimental Setup

We train three types of semantic dependency parsing models: no syntactic features (N), features extracted from gold-standard syntax trees (G), and features extracted from predicted syntax trees (P). The models of type N serve as our baseline and perform on par with the parser of Dozat and Manning (2018). For models of type G we use gold trees as inputs both during training and at test time; for models of type P, at test time we instead feed the parser with trees predicted by our syntactic parser. For the EWT models we use predicted trees during both training and testing.

Dataset	Our parser		StanfordNLP		UDPipe
	id	ood	id	ood	
SB	93.2	89.9	94.2	90.9	
DT	94.0	90.3	94.9	91.7	
EWT	85.9				85.4

Table 3: Parsing accuracy for our syntactic models on the in-domain (id) and out-of-domain (ood) test sets for the SDP data, and the regular test set for the EWT data.

Our three syntactic models (for SB, DT, and EWT) were trained using the same architecture and specifications as the semantic models, but use the factorized approach with CLE-decoding instead. Their accuracy is reported in Table 3, with additional results from StanfordNLP (Qi et al., 2018) and UDPipe (Straka and Straková, 2017) for reference. We note that in contrast to those systems' results, ours were achieved without gold POS tags.

5 Empirical Results

The results for our semantic dependency parsing models for the three graph types in the SDP dataset are presented in Table 4. For comparison, we add results reported by Dozat and Manning (2018) and Peng et al. (2018), and emphasize for each test set the overall best-performing model.

Baseline Our baseline using no syntactic features performs comparable to the systems of Dozat and Manning (2018) and Peng et al. (2018). We note that the results reported for Dozat and Manning (2018) are for models that not only use character embeddings (which we also use), but also part-of-speech tag embeddings (which we do not use).[1] The results reported for Peng et al. (2018) are for models that use predicted syntax. The slight advantage of our baseline over the latter models suggests that the network architecture can provide the same benefits as additional syntactic information.

Contribution of Syntactic Structure Looking at the results for the models informed by gold-standard syntax, we see consistent gains in both in-domain and out-of-domain settings, with substantial improvements of 3.1 labelled F1 points for DM–DT, and 2.2 points for PAS–SB. The models informed by predicted syntax, on the other hand, do not achieve any significant improvements over the baseline, and in several cases actually perform slightly worse than it. We saw the same trend in additional experiments (not reported here) where we used predicted trees even during training. Interestingly, while the baseline and the models using predicted syntax have similar F-scores, for the latter we observe a reduction of the number of false negatives (i.e., missing arcs), but also an increase in the number of false positives (i.e., incorrectly predicted arcs). The models using EWT trees do not outperform the baseline, and we omit their further investigation from the rest of the paper.

[1] The best-performing model of Dozat and Manning (2018) additionally uses lemma embeddings.

6 Error Analysis

To gain a deeper understanding for our empirical results, we complement them with an error analysis. For space reasons we will focus our analysis on the DM models (Figures 2a–2c); however, we will also discuss results from one PAS model (Figure 2d).

6.1 Error Types

Our analysis is based on a two-dimensional classification of errors. In the first dimension, for each semantic parsing model M, we break down all errors relative to the no-syntax baseline model into the following four types:

1. false negatives of the baseline avoided by M
2. false positives of the baseline avoided by M
3. false negatives of M avoided by the baseline
4. false positives of M avoided by the baseline

Type 1 thus consists of arcs that the baseline incorrectly does not and the syntax-informed model M correctly does predict, and so on.

In the second dimension, for each error type we distinguish four different sub-categories, based on the correspondence between the incorrectly predicted/not predicted arc $i \rightarrow j$ in the semantic graph and the structural relation between the head i and dependent j in the syntactic dependency tree:

(a) the dependency tree has an arc $i \rightarrow j$

(b) the dependency tree has an arc $j \rightarrow i$

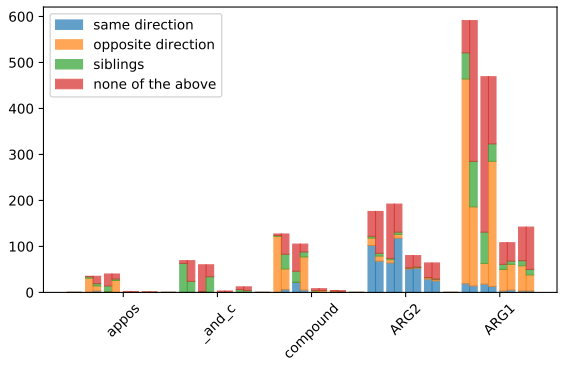
(c) i and j are siblings in the dependency tree

(d) none of the above

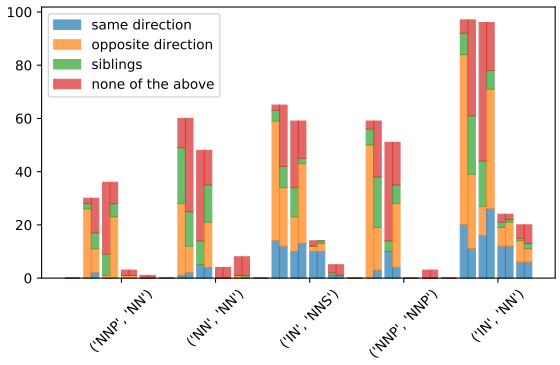
This subclassification allows us to see if there are systematic correspondences between semantic and syntactic relations, and to assess the impact of syntactic features on the semantic parser's ability to handle *difficult* arcs.

Model		DM		PAS		PSD	
		id	ood	id	ood	id	ood
N – no syntax (baseline)		92.1	87.4	92.6	89.7	79.7	77.3
DT	G – gold syntax	**95.2**	**90.9**	93.2	90.3	80.0	78.3
	P – predicted syntax	92.2	89.3	92.4	88.9	79.5	77.5
SB	G – gold syntax	92.7	88.1	**94.8**	**92.1**	80.4	**79.0**
	P – predicted syntax	92.0	87.0	92.4	88.9	79.6	77.2
EWT	P – predicted syntax	91.8	87.1	92.7	89.3	79.6	77.3
Dozat and Manning (2018)		92.7	87.8	94.0	90.6	**80.5**	78.6
Peng et al. (2018)		91.6	86.7			78.9	77.1

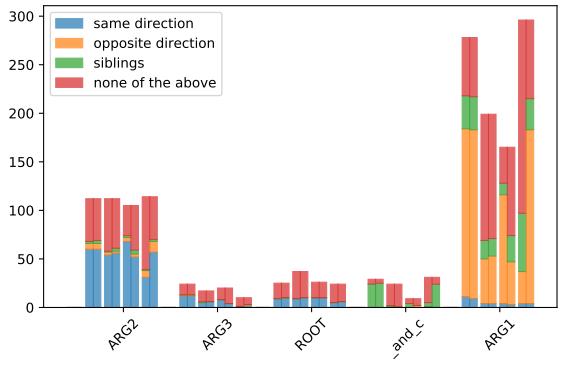
Table 4: Labeled F1 for our semantic parsers on the in-domain (id) and out-of-domain (ood) test sets.



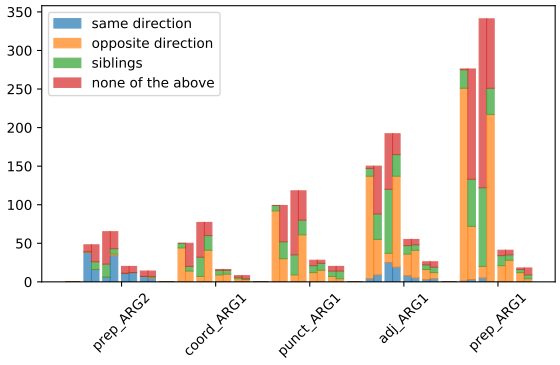
(a) DM + gold DT



(b) DM + gold DT



(c) DM + predicted DT



(d) PAS + gold SB

Figure 2: Error analysis for various models when informed by syntactic features, broken down by arc type and PoS pair. The four columns represent the four error types of Section 6.1 and colour distributions according to gold and predicted syntax features.

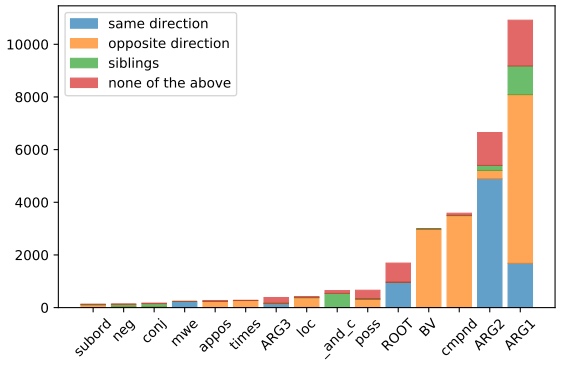


Figure 3: Relation between DM arcs and syntactic features extracted from gold-standard DT trees, broken down by arc type.

To show how syntactic heads and semantic arcs correlate independently of the parsing model, we add Figure 3. As we can see, arcs denoting the first argument of a semantic predicate (*ARG1*) occur mostly with syntactic arcs in the opposite direction. Similarly, a syntactic arc in the same direction often accompanies arcs denoting the second argument of a semantic predicate (*ARG2*); two instances from the example analysis in Figure 1 are *"take"* → *"hand"* and *"show"* → *"me"*. Another example of a clear pattern is with *compound* and conjunction (*_and_c*). Both types of arcs connect mostly nouns, but whereas *compound* arcs correlate with opposite syntactic arcs, head and dependent in a conjunction more typically share the same syntactic head (the conjunction *"and"*).

6.2 Results

We now explain how to read the graphs with the results of our two-dimensional error analysis for the DM parsing models. Figure 2a shows the outcome of this analysis when the model is informed by syntactic features extracted from gold-standard DT trees and evaluated on the development data. More specifically, it shows results for those five DM arc types for which adding syntax has the greatest effect on the absolute difference of mistakes relative to the baseline. For each arc type we plot four pairs of bars, one for each of the error types 1–4, from left to right. Figure 2b breaks down results by head–dependent part-of-speech pairs instead. Figure 2c compares the baseline with the model using predicted syntax, also on DM graphs and DT trees. Note that, as the ordering follows the absolute *difference* of mistakes, the arc types shown in Figure 2c are not the same as the ones in Figure 2a.

In all plots, the two bars in each of the four pairs of bars show the (colour-coded) distributions of the types (a)–(d) relative to gold-standard syntax (first bar) and predicted syntax (second bar); thus the distribution relative to the syntactic information actually used by each model is in the first bar in Figures 2a–2b (models informed by gold-standard syntax), and in the second bars in Figure 2c (models informed by predicted syntax).

6.3 Relation between Syntax and Semantics

To directly compare how gold-standard and predicted syntax relate to semantic arcs, we look at the differently coloured sub-columns in Figures 2a and 2b. We recall that these figures compare the no-syntax baseline to a model informed by syntactic features extracted from gold-standard analyses.

The errors avoided by the syntax-informed model (columns 1 and 2) have similar syntactic-head distributions as the general distribution in Figure 3 when looking at gold syntax. The distribution for predicted syntax does not match, due to wrong predictions when parsing these related substructures. For example, in the predicted syntax, much fewer of the avoided false negative *ARG1* arcs have syntactic arcs in the opposite direction, and instead many more of the false positives (error type 2). Using a syntactic arc in the opposite direction as an indicator, fewer false negatives would have been avoided and more false positives would have been predicted. This shows that the baseline system and the syntactic parser have difficulties analyzing the same substructures.

In Section 6 we stated that the model informed by predicted syntax increases recall at the cost of precision. This is illustrated by Figure 2c, and most pronounced for the *ARG1* type, where the number of baseline false negatives avoided by the syntax-informed model (column 1) is almost as large as the number of model false positives avoided by the baseline (column 4). The error types (a)–(d) follow the same distribution when the syntax-enhanced model improves over the baseline, but diverge when not. This finding suggests that, unsurprisingly, syntactic information helps when it is correct, and interferes otherwise. An improved system would ideally know when to trust its predicted syntax and when to rather fall back on the baseline prediction.

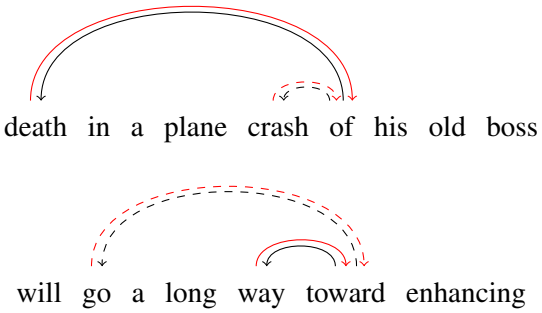


Figure 4: Graph fragments for sentences #22013118 and #22012010. Semantic arcs are shown in black, syntactic arcs in red. False negatives are drawn with continuous lines, false positives are drawn with dashed lines.

6.4 Where can Syntax help?

In order to better understand where syntax helps the semantic dependency parser, we look at the part-of-speech pairs of head and dependent in Figure 2b. The five pairs which receive the largest improvement when informed by gold-standard syntax can be divided into two groups:

(i) arcs between prepositions (IN) and nouns (NN, NNS)

(ii) arcs between multiple (proper) nouns (NN, NNS, NNP, NNPS)

Prepositional Attachment The arcs in group (i), between prepositions and nouns, represent the majority of baseline errors not only in the case of DM/DT but also in the case of PAS/SB, where they correspond (roughly) to the arc type *prep_ARG1* in Figure 2d. Figure 4 shows two graph fragments with arcs from this group; these fragments happen to be identical for DM/DT and for PAS/SB. In the two examples the gold-standard syntactic arc goes into the opposite direction than the semantic arc (our type b), and this regularity seems to be learned by the models informed by gold-standard syntax to such a degree that when the syntactic arc is incorrectly predicted, the parser also makes a corresponding mistake on the semantic side: In the examples, the prepositional phrase *"of his old boss"* is wrongly attached to the neighbouring *"plane crash"*, while *"toward"* is attached to the distant predicate *"use"* instead of the neighbouring *"way"*. Examples of this kind seem to suggest that having access to gold-standard syntax essentially 'solves' the prediction problem on the semantic side.

Compounds and Conjuncts The arcs in group (ii) include both compounds and conjuncts. The difficulties with compounds lie in determining which token is the governing head of the complete phrase and deciding which tokens are part of the compound. They naturally appear as part of conjunctions as well, where the difficulty of correctly identifying the heads of the two conjuncts is the same as identifying the heads of the compounds themselves. Similar to what we observed for preposition–noun arcs, having the governing head basically given by a syntactic arc, essentially eliminates the problem for compounds and conjuncts.

The example in Figure 5 showcases how failing to recognize a compound results in a cascade of follow-up mistakes. While *"guerrilla action"* is recognized as a compound, *"siege tactics"* is not. The word *"tactics"* is left out of the compound and therefore also the conjunct, receiving *"use"* as its syntactic head instead. This leads the semantic parser to not only fail to analyse the second compound and hence also the conjunct, but also attaching *"tactics"* as second argument (*ARG2*) to the predicate *"use"*, instead of the actual head of the complete phrase, *"action"*.

6.5 Where can Syntax not help?

While syntactic information appears to help the semantic parser in some cases, there are similar examples where syntax does not seem to be able to help at all, two of which are shown in Figure 6.

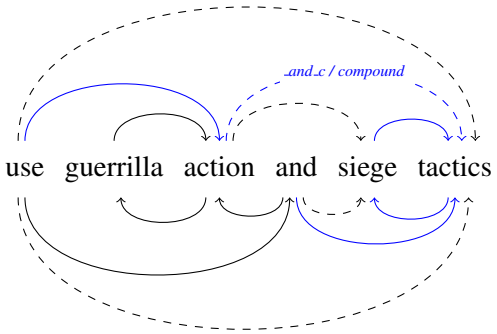


Figure 5: Graph fragments for sentence #22052046. Semantic arcs are shown above, syntactic arcs below the words. False negatives are shown in blue, false positives dashed. The blue dashed arc is an incorrectly labelled arc, annotated with the correct and the predicted label.

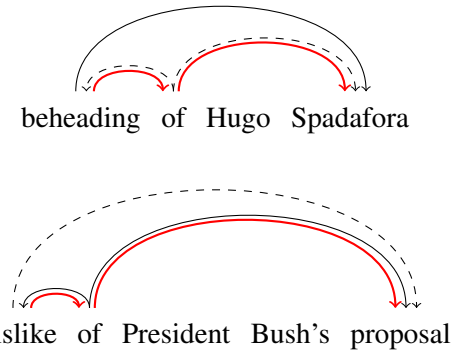


Figure 6: Graph fragments for sentences #22013141 and #22030001. Semantic arcs are shown in black, syntactic arcs in red. False negatives are drawn with continuous lines, false positives are drawn with dashed lines.

Both graph fragments contain a nominalized verb (*"beheading"* and *"dislike"*) followed by *"of"* and the object of the nominalization (*"Spadafora"* and *"proposal"*). In both cases, the parser has access to gold-standard syntax that connects the nominalization, the preposition and the object left to right. In the first instance, the parser interprets the nominalization and the object both as arguments of *"of"*, instead of directly attaching the object as an argument to the nominalization. In the second instance, the intended and observed behaviours are switched.

7 Conclusion

Our re-implementation of the state-of-the-art semantic dependency parsing architecture of Dozat and Manning (2018) performs on par with that system. Surprisingly, adding syntactic features to the standard lexical and morphological embeddings does not generally increase parsing accuracy. More specifically, while gold-standard syntactic information is highly beneficial, yielding accuracies significantly above the state of the art, adding predicted syntax does not lead to consistent improvements.

Our error analysis shows that there is some overlap of the information that syntactic dependency trees and semantic dependency graphs encode, in the sense that both tend to mirror each other. We have provided examples for cases that are difficult to analyze due to their inherent ambiguity. In some cases, these examples suggest that adding gold-standard syntax essentially also reveals the correct semantic analysis to the parser. This means that high-precision syntactic parsing holds significant promises even for semantic parsing, but our experiments suggest that the state of the art in syntactic dependency parsing may still be too low to fully capitalize on this potential. We believe however, that a joint syntactic–semantic parser that is able to dynamically leverage both structures (trained, perhaps, using a multi-task objective), would be an opportunity for further advances in both syntactic and semantic dependency parsing.

Acknowledgements

We are grateful for the computational resources provided by Sigma2 in Oslo through the Nordic e-Infrastructure Collaboration (NeIC) and the Nordic Language Processing Laboratory (NLPL, `www.nlpl.eu`).

References

Mariana S. C. Almeida and André F. T. Martins. 2015. Lisbon: Evaluating TurboSemanticParser on Multiple Languages and Out-of-Domain Data. *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 970–973.

Junjie Cao, Sheng Huang, Weiwei Sun, and Xiaojun Wan. 2017. Parsing to 1-Endpoint-Crossing, Pagenumber-2 Graphs. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1:2110–2120.

Yoeng-Jin Chu and Tseng-Hong Liu. 1965. On the shortest arborescence of a directed graph. *Scientia Sinica*, 14:1396–1400.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. The Stanford Typed Dependencies Representation. In *CF+CDPE@COLING*, pages 1–8. Coling 2008 Organizing Committee.

Timothy Dozat and Christopher D. Manning. 2018. Simpler but More Accurate Semantic Dependency Parsing. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2:484–490.

Jack Edmonds. 1967. Optimum Branchings. *Journal of Research of the national Bureau of Standards B*, 71(4):233–240.

Dan Flickinger, Jan Hajič, Angelina Ivanova, Marco Kuhlmann, Yusuke Miyao, Stephan Oepen, and Daniel Zeman. 2016. SDP 2014 & 2015: Broad Coverage Semantic Dependency Parsing LDC2016T10.

W. Nelson Francis and Henry Kučera. 1985. Frequency Analysis of English Usage: Lexicon and Grammar. *Journal of English Linguistics*, 18(1):64–70.

Yarin Gal and Zoubin Ghahramani. 2016. A Theoretically Grounded Application of Dropout in Recurrent Neural Networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 1027–1035. Curran Associates Inc.

Jan Hajic, Eva Hajicová, Jarmila Panevová, Petr Sgall, Ondřej Bojar, Silvie Cinková, Eva Fućiková, Marie Mikulová, Petr Pajas, Jan Popelka, Jiři Semecký, Jana Sindlerová, Jan Štěpánek, Josef Toman, Zdenka Uresová, and Zdenek Zabokrtský. 2012. Announcing Prague Czech-English Dependency Treebank 2.0. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC 2012)*, pages 3153–3160, İstanbul, Turkey. European Language Resources Association.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-term Memory. *Neural computation*, 9:1735–80.

Angelina Ivanova, Stephan Oepen, Lilja Øvrelid, and Dan Flickinger. 2012. Who Did What to Whom?: A Contrastive Study of Syntacto-semantic Dependencies. In *Proceedings of the Sixth Linguistic Annotation Workshop*, LAW VI '12, pages 2–11, Stroudsburg, PA, USA. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980.

Marco Kuhlmann and Peter Jonsson. 2015. Parsing to Noncrossing Dependency Graphs. *Transactions of the Association of Computational Linguistics*, 3(1):559–570.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Comput. Linguist.*, 19(2):313–330.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2017. Regularizing and Optimizing LSTM Language Models. *CoRR*, abs/1708.02182.

Yusuke Miyao. 2006. *From Linguistic Theory to Syntactic Analysis: Corpus-Oriented Grammar Development and Feature Forest Model*. Ph.D. thesis.

Joakim Nivre, Željko Agić, Lars Ahrenberg, Maria Jesus Aranzabe, Masayuki Asahara, Aitziber Atutxa, Miguel Ballesteros, John Bauer, Kepa Bengoetxea, Riyaz Ahmad Bhat, Eckhard Bick, Cristina Bosco, Gosse Bouma, Sam Bowman, Marie Candito, Gülşen Cebiroğlu Eryiğit, Giuseppe G. A. Celano, Fabricio Chalub, Jinho Choi, Çağrı Çöltekin, Miriam Connor, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Arantza Diaz de Ilarraza, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Marhaba Eli, Tomaž Erjavec, Richárd Farkas, Jennifer Foster, Cláudia Freitas, Kataŕına Gajdošová, Daniel Galbraith, Marcos Garcia, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökırmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Nizar Habash, Jan Hajič, Linh Hà Mỹ, Dag Haug, Barbora Hladká, Petter Hohle, Radu Ion, Elena Irimia, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşıkara, Hiroshi Kanayama, Jenna Kanerva, Natalia Kotsyba, Simon Krek, Veronika Laippala, Phuong Lê H`ông, Alessandro Lenci, Nikola Ljubešić, Olga Lyashevskaya, Teresa Lynn, Aibek Makazhanov, Christopher Manning, Cătălina Mărănduc, David Mareček, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Anna Missilä, Verginica Mititelu, Yusuke Miyao, Simonetta Montemagni, Amir More, Shunsuke Mori, Bohdan Moskalevskyi, Kadri Muischnek, Nina Mustafina, Kaili Müürisep, Luong Nguỹên Th i, Huỳên Nguỹên Th i Minh, Vitaly Nikolaev, Hanna Nurmi, Stina Ojala, Petya Osenova, Lilja Øvrelid, Elena Pascual, Marco Passarotti, Cenel-Augusto Perez, Guy Perrier, Slav Petrov, Jussi Piitulainen, Barbara Plank, Martin Popel, Lauma Pretkalniņa, Prokopis Prokopidis, Tiina Puolakainen, Sampo Pyysalo, Alexandre Rademaker, Loganathan Ramasamy, Livy Real, Laura Rituma, Rudolf Rosa, Shadi Saleh, Manuela Sanguinetti, Baiba Saulīte, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Lena Shakurova, Mo Shen, Dmitry Sichinava, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Dima Taji, Takaaki Tanaka, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Larraitz Uria, Gertjan van Noord, Viktor Varga, Veronika Vincze, Jonathan North Washington, Zdeněk Žabokrtský, Amir Zeldes, Daniel Zeman, and Hanzhi Zhu. 2017. Universal Dependencies 2.0.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinkova, Dan Flickinger, Jan Hajic, and Zdenka Uresova. 2015. SemEval 2015 Task 18: Broad-Coverage Semantic Dependency Parsing. *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 915–926.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajic, Angelina Ivanova, and Yi Zhang. 2014. SemEval 2014 Task 8: Broad-Coverage Semantic Dependency Parsing. *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 63–72.

Stephan Oepen and Jan Tore Lønning. 2006. Discriminant-Based MRS Banking. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC-2006)*, Genoa, Italy. European Language Resources Association (ELRA).

Hao Peng, Sam Thomson, and Noah A. Smith. 2017. Deep Multitask Learning for Semantic Dependency

Parsing. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1:2037–2048.

Hao Peng, Sam Thomson, and Noah A. Smith. 2018. Backpropagating through Structured Argmax using a SPIGOT. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1863–1873, Melbourne, Australia. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Peng Qi, Timothy Dozat, Yuhao Zhang, and Christopher D. Manning. 2018. Universal Dependency Parsing from Scratch. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 160–170. Association for Computational Linguistics.

Daniel Roxbo. 2019. A Detailed Analysis of Semantic Dependency Parsing with Deep Neural Networks. Master's thesis, Linköping University, Human-Centered systems.

Milan Straka and Jana Straková. 2017. Tokenizing, POS Tagging, Lemmatizing and Parsing UD 2.0 with UDPipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99. Association for Computational Linguistics.

Daniel Varab and Natalie Schluter. 2018. UniParse: A universal graph-based parsing toolkit. *CoRR*, abs/1807.04053.

Li Wan, Matthew Zeiler, Sixin Zhang, Yann LeCun, and Rob Fergus. 2013. Regularization of Neural Networks Using Dropconnect. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pages III–1058–III–1066. JMLR.org.

Yuxuan Wang, Wanxiang Che, Jiang Guo, and Ting Liu. 2018. A Neural Transition-Based Approach for Semantic Dependency Graph Parsing. In *AAAI*, pages 5561–5568. AAAI Press.

Xun Zhang, Yantao Du, Weiwei Sun, and Xiaojun Wan. 2016. Transition-Based Parsing for Deep Dependency Structures. *Computational Linguistics*, 42(3):353–389.