Evaluating LSTM models for grammatical function labelling

Bich-Ngoc Do[♢] and **Ines Rehbein**[♣]
Leibniz ScienceCampus
Universität Heidelberg[♢] / Institut für Deutsche Sprache Mannheim[♣]
{do|rehbein}@cl.uni-heidelberg.de

Abstract

To improve grammatical function labelling for German, we augment the labelling component of a neural dependency parser with a decision history. We present different ways to encode the history, using different LSTM architectures, and show that our models yield significant improvements, resulting in a LAS for German that is close to the best result from the SPMRL 2014 shared task (without the reranker).

1 Introduction

For languages with a non-configurational word order and rich(er) morphology, such as German, grammatical function (GF) labels are essential for interpreting the meaning of a sentence. *Case syncretism* in the German case paradigm makes GF labelling a challenging task. See (1) for an example where the nouns in the sentence are ambiguous between different cases, which makes it hard for a statistical parser to recover the correct reading.

(1) Telefonkarten bleibt nichts erspart.
phone cards$_{Nom/Acc/Dat}$ remains nothing$_{Nom/Acc}$ spared.
"Phone cards are subjected to all kinds of abuse."

We approach the problem of GF labelling as a subtask of dependency parsing, where we first generate unlabelled trees and, in the second step, try to find the correct labels. This pipeline architechture gives us more flexibility, allowing us to use the labeller in combination with our parser, but also to apply it to the unlabelled output of other parsing systems without the need to change or retraining the parsers.

The approach also makes it straightforward to test different architectures for GF labelling. We are especially interested in the influence of different input structures representing different (surface versus structural) orders of the input. In particular, we compare models where we present the unlabelled tree in linear order with a model where we encode the parser output as a tree. We show that all models are able to learn GFs with a similar overall LAS, but the model where the tree is encoded in a breadth-first order outperforms all other models on labelling core argument GFs.

2 Related Work

Grammatical function labelling is commonly integrated into syntactic parsing. Few studies have adressed the issue as a separate classification task. While most of them assign grammatical functions on top of constituency trees (Blaheta and Charniak, 2000; Jijkoun and de Rijke, 2004; Chrupała and van Genabith, 2006; Klenner, 2007; Seeker et al., 2010), less work has tried to predict GF labels for unlabelled dependency trees. One of them is McDonald et al. (2006) who first generate the unlabelled trees using a graph-based parser, and then model the assignment of dependency labels as a sequence labelling task.

Another approach has been proposed by Zhang et al. (2017) who present a simple, yet efficient and accurate parsing model that generates unlabelled trees by identifying the most probable head for each token in the input. Then, in a post-processing step, they assign labels to each head-dependent pair, using a two-layer rectifier network.

Dependency Parsing as Head Selection Our labelling model is an extension of the parsing model of Zhang et al. (2017). We use our own implementation of the head-selection parser and focus on the grammatical function labelling part. The parser uses a bidirectional LSTM to extract a dense, positional representation a_i of the word w_i at position i in a sentence:

Proceedings of the 15th International Conference on Parsing Technologies, pages 128–133,
Pisa, Italy; September 20–22, 2017.

$$h_i^F = \text{LSTM}^F(x_i, h_{i-1}^F) \quad (1)$$
$$h_i^B = \text{LSTM}^B(x_i, h_{i+1}^B) \quad (2)$$
$$a_i = [h_i^F; h_i^B] \quad (3)$$

x_i is the input at position i, which is the concatenation of the word embeddings and the tag embeddings of word w_i. An artificial root token w_0 is added at the beginning of each sentence.

The unlabelled tree is then built by selecting the most probable head for each word. The score of word w_j being the head of word w_i is computed by a single hidden layer neural network on their representations a_j and a_i.

An additional classifier with two rectified hidden layers is used to predict dependency labels, and is trained separately from the unlabeled parsing component, in a pipeline architecture. The classifier predictions are based on the representations of the head and the dependent, b_j and b_i, which are the concatenation of the input and the bidirectional LSTM-based representations:

$$b_i = [x_i; a_i] \quad (4)$$

Despite its simplicity and the lack of global optimisation, Zhang et al. (2017) report competitive results for English, Czech, and German.

3 Labeling Dependencies with History

Although the labelling approach in Zhang et al. (2017) is simple and efficient, looking at head and dependent only when assigning the labels comes with some disadvantages. First, some labels are easier to predict when we also take context into account, e.g. the parent and grandparent nodes or the siblings of the head or dependent.

Consider, for example, the following sentence: *Is this the future of chamber music?* and its syntactic structure (figure 1). If we only consider the nodes *this* and *future*, there is a chance that the edge between them is labelled as *det* (determiner). However, if we also look at the local context, we know that node *the* to the left of *future* is more likely to be the determiner, and thus *this* should be assigned a different label.

Second, when looking at the parser output, we notice some errors that are well-known from other local parsing models, such as the assignment of duplicate subjects for the same predicate. To address this issue, we propose an extended labelling

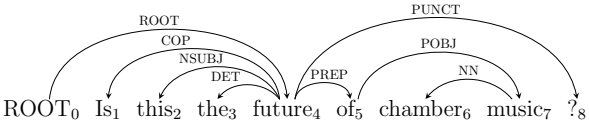

Figure 1: The dependency tree of the sentence *Is this the future of chamber music?*

model that incorporates a decision history. To that end, we design different LSTM architectures for the labelling task and compare their performance on German, Czech and English.

Label prediction as a sequence labelling task Presenting the input to the labeller in sequential surface order does not seem very intuitive when we want to assign labels to a tree. This approach, however, was adapted by McDonald et al. (2006). In their work, they consider all dependents $x_{j1}, ..., x_{jM}$ of a head x_i and label those edges $(i, j1), ..., (i, jM)$ in a sequence.

We argue, however, that it is not enough to know the labels of the siblings, but that we also need to consider nodes at different levels in the tree. Therefore, when predicting the label for the current node, we consider all label decisions in the history, and feed them to a bidirectional LSTM. Given a sequence of nodes $S = (w_1, ..., w_N)$ and their corresponding head $(h_1, ..., h_N)$, at each recurrent step, we input the learned representation of the head and the dependent:

$$h_i^{F(\text{lbl})} = \text{LSTM}_{\text{lbl}}^F(b_i, b_{h_i}, h_{i-1}^{F(\text{lbl})}) \quad (5)$$
$$h_i^{B(\text{lbl})} = \text{LSTM}_{\text{lbl}}^B(b_i, b_{h_i}, h_{i+1}^{B(\text{lbl})}) \quad (6)$$

After that, the concatenated hidden states $[h_i^{F(lbl)}; h_i^{B(lbl)}]$ are projected to a softmax layer to predict the label.

When presenting a tree as a sequence, we experiment with two different input orders:

- **BILSTM(L)**: Tree nodes are ordered according to their surface order in the sentence (linear order; figure 2).

- **BILSTM(B)**: Tree nodes are ordered according to a breadth-first traversal (BFS) of the tree, starting from the root node. Here, siblings are closer to each other in the history.

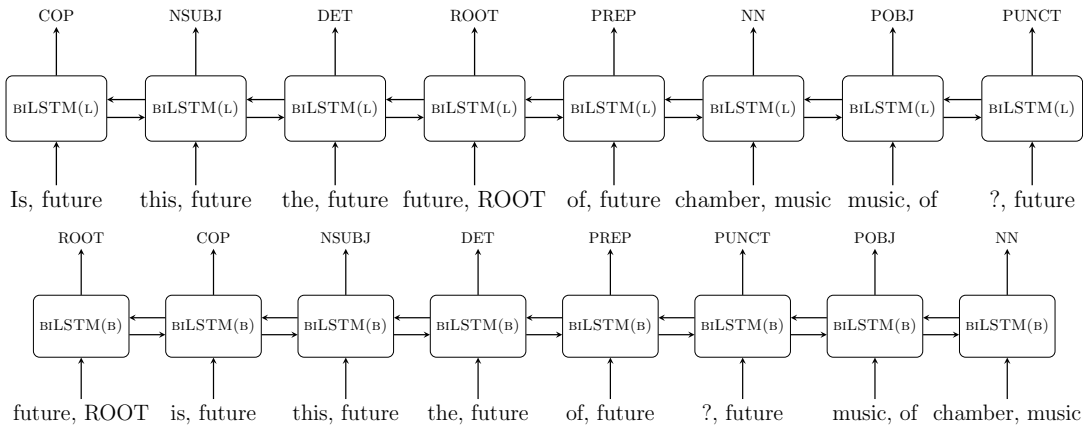

Figure 2: The processing order of the sentence in figure 1 a) in the BILSTM(L) model (top) and b) in the BILSTM(B) model (bottom).

Top-down tree LSTM Intuitively, it seems more natural to present the input as a tree structure when trying to predict the dependency labels. We do that by adopting the top-down tree LSTM model (Zhang et al., 2016) that processes nodes linked through dependency paths in a top-down manner. To make it comparable to the previous LSTM models, we only use *one LSTM* instead of four, and do not stack LSTMs. The hidden state is computed as follow:

$$h_i^{(lbl)} = \text{treeLSTM}(b_i, h_{i-1}) \quad (7)$$

After that, we proceed as we did for the **BILSTM** models (see above). Note that the processing order i is also the BFS order. We call this model **TREELSTM** (figure 3).

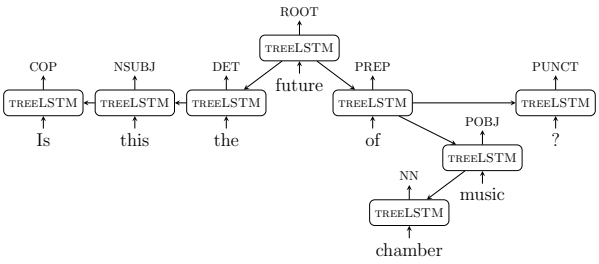

Figure 3: The processing order of the sentence in figure 1 in the TREELSTM model.

4 Experiments

Our interest is focussed on German, but to put our work in context, we follow Zhang et al. (2017) and report results also for English, which has a configurational word order, and for Czech, which has a free word order, rich morphology, and less ambiguity in the case paradigm than German.

For English, we use the Penn Treebank (PTB) (Marcus et al., 1993) with standard training/dev/test splits. The POS tags are assigned using the Stanford POS tagger (Toutanova et al., 2003) with ten-way jackknifing, and constituency trees are converted to Stanford basic dependencies (De Marneffe et al., 2006). The German and Czech data come from the CoNLL-X shared task (Buchholz and Marsi, 2006) and our data split follows Zhang et al. (2017). As the CoNLL-X testsets are rather small ($\sim$ 360 sentences), we also train and test on the much larger German SPMRL 2014 shared task data (Seddah et al., 2014) (5,000 test sentences). For the SPMRL data we use the predicted POS tags provided by the shared task organisers.

4.1 Setup

We test different labelling models on top of the unlabelled trees produced by our re-implementation of the *parsing as head selection* model (§2).

We first train the unlabelled parsing models for the three languages. Unless stated otherwise, all parameters are set according to Zhang et al. (2017), and tag embedding size was set to 40 for all languages. Please note that we do *not* use pre-trained embeddings in our experiments.

In the next step, we train four different labelling models: the labeller of Zhang et al. (2017) that uses a rectifier neural network with two hidden layers (**baseline**), two bidirectional LSTM models (**BILSTM(L)** and **BILSTM(B)**), and one tree LSTM model (**TREELSTM**) (§3).

The hidden layer dimension in all LSTM models was set to 200. The models were trained for 10 epochs, and were optimized using Adam

Model	en	cs	de$_{CoNLL}$	de$_{SPMRL}$
UAS	93.35	89.70	93.09	91.29
Baseline	91.58	83.42	90.22	88.15
BILSTM(L)	**91.92***	**84.08***	90.87*	88.73*
BILSTM(B)	91.91*	83.80	**90.97***	**88.74***
TREELSTM	**91.92***	83.82	90.89*	**88.74***
DENSE	91.90	81.72	89.60	-

Table 1: Results for different labellers applied to the unlabelled parser output. The first row reports UAS for the input to the labellers. The last row (DENSE) shows the results from Zhang et al. (2017). (*) indicates that the difference between the model and the baseline is statistically significant ($p < .001$).

(Kingma and Ba, 2015) with default parameters (initial learning rate 0.001, first momentum coefficient 0.9, second momentum coefficient 0.999). We used L2 regularization with a coefficient of 10^{-3} and max-norm regularization with an upper bound of 5.0. The dropout (Srivastava et al., 2014) rate was set to 0.05 for the input connections, and 0.5 for the rest.

4.2 Results

Table 1 shows the unlabelled attachment score (UAS) for the unlabelled trees and the labelled attachment scores (LAS) for the different labellers (excluding punctuation). All history-based labelling models perform significantly better than the local baseline model,[1] but for English the improvements are smaller (0.3%) than for the non-configurational languages (∼0.7%).

While we tried to reimplement the model of Zhang et al. (2017) following the details in the paper, our reimplemented model yields higher scores for German, compared to the results in the paper. The scores for English are slightly lower since, in contrast to Zhang et al. (2017), we do not use pre-trained embeddings. When using our history-based labellers, we get similar results for English (91.9%) and higher results for both Czech (84.1% vs. 81.7%) and German (91.0% vs. 89.6%) on the same data *without* using pre-trained embeddings or post-processing.

On the SPMRL 2014 shared task data, our results are only 0.3% lower than the ones of the winning system (Björkelund et al., 2014) *without* reranker (*blended*).[2] To further illustrate the effectiveness of our models, we also ran our labeller on the *unlabelled output* of the SPMRL 2014 winning system and on *unlabelled gold* trees. On the output of the *blended* system LAS slightly improves from 88.62% to 88.76% (TREELSTM).[3] When applied to *unlabelled gold* trees, the distance between our models and the baseline becomes larger and the best of our history-based models (BILSTM(B), 97.38%) outperforms the original labeller of Zhang et al. (2017) (96.15%) by more than 1%.

de$_{SPMRL}$	SB	OA	DA	PD
	# 6,638	# 3,184	# 568	# 1,045
baseline	90.3	83.6	64.7	77.1
BILSTM(L)	91.4	85.3	67.7	80.0
BILSTM(B)	**91.9**	**85.4**	**69.3**	**80.5**
treeLSTM	91.2	85.1	68.6	79.8
de$_{SPMRL}$	**AG**	**PG**	**OC**	**OG**
	# 2,241	# 388	# 3,652	# 21
baseline	91.3	80.0	90.1	0
BILSTM(L)	91.3	81.6	90.5	16.0
BILSTM(B)	**91.5**	**82.4**	**90.7**	**37.0**
treeLSTM	91.4	81.4	90.2	27.6

Table 2: LAS for core argument functions (German SPMRL data), and frequency (#) of GF in the testset (SB: subj, OA: acc.obj, DA: dat.obj, PD: pred, AG: gen.attribute, PG: phrasal genitive, OC: clausal obj, OG: gen.obj).

We would like to emphasize that our history-based LSTM labeller is practically simple and computationally inexpensive (as compared to global training or inference), so our model manages to preserve simplicity while significantly improving labelling performance.

4.3 Discussion

Most strikingly, all three models seem to perform roughly the same, and the TREELSTM model fails to outperform the other two models. However, in comparison to the BILSTM models, the TREELSTM model has a smaller number of parameters, and the history only flows in one direction. The tree model also has a shorter history chain since nodes are linked by paths from the root (figure 3), which might explain why it does not yield better results than the linear LSTM models.

The overall results suggest that the order in which the nodes are presented in the history does not have any impact on the labelling results. However, when looking at results for individual core argument functions (subject, direct object, etc.), a

[1] For significance testing, we use Bikel's Randomized Parsing Evaluation Comparator (http://www.cis.upenn.edu/~dbikel/software.html).

[2] The shared task winner is a complex ensemble system that generates a tree by *blending* the output of three parsers (Mate, Turbo, BestFirst; see (Björkelund et al., 2014)).

[3] Following Björkelund et al. (2014), here we include punctuation in the evaluation.

	GF	en	cs	de$_{SPMRL}$
dep-length	sb	3.1	3.4	3.9
	dobj	2.5	*2.4	4.2
	iobj	1.7	-	4.7
left-head ratio	sb	4.6	32.5	34.2
	dobj	97.4	*77.5	37.2
	iobj	100.0	-	27.5

Table 3: Avg. dependency length and ratio of left arcs vs. all (left + right) arc dependencies for args. (* in the Czech data, *Obj* subsumes all types of objects, not only direct objects)

more pronounced pattern emerges (table 2).[4] Here we see the benefit of encoding the siblings close to each other in the history: For all core argument functions, the BILSTM(B) model outperforms the other models.

To find out why the history-based models work better for Czech and German than for English, we compared the average dependency length as well as the variability in head direction (how often e.g. the head of a subject is positioned to the left, in relation to the total number of subjects). Table 3 suggests that the success of the history-based models is not due to a better handling of long dependencies but that they are better in dealing with the uncertainty in head direction (also see Gulordava and Merlo (2016)).

5 Conclusions

We have shown that GF labelling, which is of crucial importance for languages like German, can be improved by combining LSTM models with a decision history. All our models outperform the original labeller of Zhang et al. (2017) and give results in the same range as the best system from the SPMRL-2014 shared task (without the reranker), but with a much simpler model. Our results show that the history is especially important for languages that show more word order variation. Here, presenting the input in a structured BFS order not only significantly outperforms the baseline, but also yields improvements over the other LSTM models on core grammatical functions.

Acknowledgments

We would like to thank Minh Le for his help with data pre-processing. This research has been conducted within the Leibniz Science Campus "Empirical Linguistics and Computational Modeling", funded by the Leibniz Association under grant no. SAS-2015-IDS-LWC and by the Ministry of Science, Research, and Art (MWK) of the state of Baden-Württemberg.

[4]We evaluate GFs on the German SPMRL data which are sufficiently large with 5,000 test sentences. The CoNLL datasets, in comparison, only include ∼360 test sentences.

References

Anders Björkelund, Özlem Çetinoğlu, Agnieszka Faleńska, Richárd Farkas, Thomas Müller, Wolfgang Seeker, and Zsolt Szántó. 2014. Introducing the IMS-Wroclaw-Szeged-CIS entry at the SPMRL 2014 Shared Task: Reranking and Morpho-syntax meet Unlabeled Data. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*. Dublin City University, Dublin, Ireland, pages 97–102. http://www.aclweb.org/anthology/W14-6110.

Don Blaheta and Eugene Charniak. 2000. Assigning function tags to parsed text. In *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference*. NAACL '00, pages 234–240.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, Stroudsburg, PA, USA, CoNLL-X '06, pages 149–164.

Grzegorz Chrupała and Josef van Genabith. 2006. Using machine-learning to assign function labels to parser output for Spanish. In *Proceedings of the COLING/ACL on Main Conference Poster Sessions*. COLING-ACL '06, pages 136–143.

Marie-Catherine De Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *The fifth international conference on Language Resources and Evaluation*. LREC'06, pages 449–454.

Kristina Gulordava and Paola Merlo. 2016. Multilingual dependency parsing evaluation: a large-scale analysis of word order properties using artificial data. *TACL* 4:343–356.

Valentin Jijkoun and Maarten de Rijke. 2004. Enriching the output of a parser using memory-based learning. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*. pages 311–318.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *The International Conference on Learning Representations*. San Diego.

Manfred Klenner. 2007. Shallow dependency labeling. In *Proceedings of the 45th Annual Meeting of the*

ACL on Interactive Poster and Demonstration Sessions. ACL '07, pages 201–204.

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

Ryan McDonald, Kevin Lerman, and Fernando Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of the 10th Conference on Computational Natural Language Learning*. CoNLL-X '06, pages 206–210.

Djamé Seddah, Sandra Kübler, and Reut Tsarfaty. 2014. Introducing the SPMRL 2014 shared task on parsing morphologically-rich languages. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*. Dublin City University, Dublin, Ireland, pages 103–109. http://www.aclweb.org/anthology/W14-6111.

Wolfgang Seeker, Ines Rehbein, Jonas Kuhn, and Josef van Genabith. 2010. Hard constraints for grammatical function labelling. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. ACL '10, pages 1087–1097.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15:1929–1958. http://jmlr.org/papers/v15/srivastava14a.html.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*. NAACL '03, pages 173–180.

Xingxing Zhang, Jianpeng Cheng, and Mirella Lapata. 2017. Dependency parsing as head selection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*. EACL'17, pages 665–676.

Xingxing Zhang, Liang Lu, and Mirella Lapata. 2016. Top-down tree long short-term memory networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 310–320. http://www.aclweb.org/anthology/N16-1035.