Neural Reranking for Dependency Parsing: An Evaluation

Bich-Ngoc Do[♢], **Ines Rehbein**[♣]
Leibniz ScienceCampus[♢], Data and Web Science Group[♣]
Universität Heidelberg, Heidelberg, Germany[♢]
Universität Mannheim, Mannheim, Germany[♣]
do@cl.uni-heidelberg.de, ines@informatik.uni-mannheim.de

Abstract

Recent work has shown that neural rerankers can improve results for dependency parsing over the top k trees produced by a base parser. However, all neural rerankers so far have been evaluated on English and Chinese only, both languages with a configurational word order and poor morphology. In the paper, we re-assess the potential of successful neural reranking models from the literature on English and on two morphologically rich(er) languages, German and Czech. In addition, we introduce a new variation of a discriminative reranker based on graph convolutional networks (GCNs). We show that the GCN not only outperforms previous models on English but is the only model that is able to improve results over the baselines on German and Czech. We explain the differences in reranking performance based on an analysis of a) the gold tree ratio and b) the variety in the k-best lists.

1 Introduction

Neural models for dependency parsing have been a tremendous success, pushing state-of-the-art results for English on the WSJ benchmarking dataset to over 94% LAS (Dozat and Manning, 2017). Most state-of-the-art parsers, however, are local and greedy and are thus expected to have problems finding the best *global* parse tree. This suggests that combining greedy, local parsing models with some mechanism that adds a global view on the data might increase parsing accuracies even further.

In this work, we look into incorporating global information for dependency parsing via *reranking*. Different model architectures have been proposed for neural reranking of dependency parse trees (Le and Zuidema, 2014; Zhu et al., 2015; Zhou et al., 2016). Despite achieving modest or even substantial improvements over the baseline parser, however, all the systems above only report performance on English and Chinese data, both morphologically poor languages with a configurational word order and mostly projective tree structures.

In the paper, we thus try to reproduce results for different reranking models from the literature on English data and compare them to results for German and Czech, two morphologically rich(er) languages (MRLs) with a high percentage of non-projective structures. In addition, we present a new discriminative reranking model based on graph convolutional networks (GCNs). Our GCN reranker outperforms the other rerankers on English and is also the only model able to obtain small improvements over the baseline parser on German and Czech while the other rerankers fail to beat the baselines. The improvements, however, are not significant and raise the question what makes neural reranking of MRLs more difficult than reranking English or Chinese.

We analyze the differences in performance on the three languages and show that the reasons for this failure are due to the composition and quality of the k-best lists. In particular, we show that the *gold tree ratio* in the English k-best list is much higher than for German and Czech, and that the trees in the English k-best list show a higher *variety*, thus making it easier for the reranker to distinguish between high- and low-quality trees.

The paper is structured as follows. In §2, we review related work on reranking for neural dependency parsing. The different reranking models are described in detail in §3. In §4, we first reproduce reranking results for English and evaluate our new reranker on the English data. Then we test the different models on the two morphologically rich(er) languages and present the results of our evaluation and our analysis, before we conclude in §5.

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 4123–4133

2 Related Work

Reranking is a popular technique to improve parsing performance of the output of a base parser. First, the top k candidate trees are generated by the base parser, then these trees are reranked using additional features not accessible to the base parser. This adds a more global and complete view of the trees, in contrast to the local and incomplete features used by the parser.

Discriminative rerankers have been a success story in constituency parsing (Collins and Koo, 2005; Charniak and Johnson, 2005). A disadvantage of the traditional feature-rich rerankers is that the large number of potentially sparse features makes them prone to overfitting, and also reduces the efficiency of the systems. Neural rerankers offer a solution to that problem by learning dense, low-dimensional feature representations that are better at generalization, and so reduce the risk of overfitting.

Neural reranking The first neural reranker has been presented by Socher et al. (2013) for constituency parsing, based on a recursive neural network which processes the nodes in the parse tree bottom-up and learns dense feature presentations for the whole tree. This approach was adapted for dependency parsing by Le and Zuidema (2014). Zhu et al. (2015) improve on previous work by proposing a recursive convolutional neural network (RCNN) architecture for reranking which can capture syntactic and semantic properties of words and phrases in the parse trees (see §3 for a more detailed description of the two models).

k-best vs. forest reranking There exist two different approaches to reranking for parsing: k-best reranking and forest reranking. In k-best reranking, the complete parse tree is encoded and presented to the reranker. A disadvantage of k-best reranking is the limited scope of the k-best list which provides an upper bound for reranking performance. In contrast, a packed parse forest is a compact representation of exponentially many trees of which each node represents a deductive step. Forest reranking (Huang, 2008; Hayashi et al., 2013) approximately decodes the highest scored tree with both local and non-local features in a parse forest with cube pruning (Huang and Chiang, 2005).

In our work, we focus on neural reranking of a k-best list of parses generated by a base parsing system as we could not find any available parsers that are both non-projective and produce packed parse forests as output.

3 Neural Reranking Models for Dependency Parsing

In this section, we look into reranking for dependency parsing and compare two different types of models: the *generative* inside-outside recursive neural network (**IORNN**) reranker (Le and Zuidema, 2014) and the *discriminative* reranker based on recurrent convolutional neural networks (**RCNNs**) (Zhu et al., 2015). In addition, we propose a new reranking model for dependency parsing that employs graph convolutional networks (**GCNs**) to encode the trees.

3.1 Generative models

A generative reranking model scores a dependency structure by estimating its generation probability. The probability of generating a fragment of a dependency tree (e.g., a node) D depends on its dependency context C_D. The amount of information used in C_D is called the *order* of the generative model. Ideally, we want to generate a dependency subtree D based on ∞-order context C_D^∞ which includes all ancestors of D, their siblings, and all siblings of D. As the ∞-order counting model is impracticable due to data sparsity, Le and Zuidema (2014) propose the IORNN model to encode the context to generate each node in a dense vector.

IORNN The IORNN extends the idea of recursive neural networks (Socher et al., 2010) for constituent parsing where the *inner representation* of a node is computed bottom up. It also adds a second vector to each node, an *outer representation*, which is computed top down. The inner representation represents the content of the subtree at the current node, while the outer representation represents the context used to generate that node. The model is further adapted to ∞-order dependency trees with *partial outer representations* that represent the partial context while generating dependents from left to right. For details on how to compute these representations, please refer to Le and Zuidema (2014).

Training The IORNN is trained to maximize the probability of generating each word w given its partial outer representation $\bar{\mathbf{o}}_w$:

$$\mathcal{L}(\Theta) = \frac{1}{m} \sum_{T \in D} \sum_{w \in T} \log P(w|\bar{\mathbf{o}}_w) \qquad (1)$$

where D is the set of dependency trees, and m is the total number of words.

3.2 Discriminative models

In contrast to generative models, a discriminative reranker learns to distinguish the correct parse tree of a sentence from the incorrect ones. Since the tree space is huge, one cannot generate all possible trees to train the model, but can only use a subset of the trees generated by the base parser. Therefore, a discriminative reranker is only optimized for one specific parser and can easily overfit the error types of the k-best list. The common idea of all models in this section is to encode the structure of a dependency tree via its node and/or edge representations. Node representations are computed either recursively bottom-up (**RCNN**) or in a step-by-step recurrent manner (**GCN**).

RCNN A RCNN *recursively* encodes each subtree with regards to its children using a *convolutional* layer. At each dependency node h, a RCNN module computes its hidden representation $\mathbf{h}$ and a plausibility score $s(h)$ based on the representation of its children. For details, see Zhu et al. (2015).

Given a sentence x and its dependency tree y, the score of y is computed by summing up the scores of all inner nodes h:

$$s(x, y, \Theta) = \sum_{h \in y} s(h) \tag{2}$$

The network then outputs the predicted tree $\hat{y}$ from the input list $\mathbf{gen}(x)$ with the highest score:

$$\hat{y} = \text{argmax}_{y \in \mathbf{gen}(x)} s(x, y, \Theta) \tag{3}$$

The bottom-up fashion used in the RCNN can cause disproportion between the tree structure and its representation due to the order in the recursive computation. Consider two trees that only differ in one edge. Their node representations will be more similar if the edge appears higher up in the tree and less so if the edge is closer to the lower level, since the difference spreads to the upper level. Thus, we believe that a discriminative reranker can benefit from a model that considers nodes in a tree more equally, as done in our GCN model below.

GCN GCNs have been used to encode nodes in a graph with (syntactic) information from their neighbors. By stacking several layers of GCNs, the learned representation can capture information about directly connected nodes (with only one layer), or nodes that are K hops away (with K layers). We adapt the syntactic gated GCNs for semantic role labeling from Marcheggiani and Titov (2017) to encode the parse trees in our experiments. To our best knowledge, this is the first time GCNs are used for reranking in dependency parsing.

Let the hidden representation of node v after K GCN layers be $\mathbf{h}_v^{(K)}$. The plausibility score of each tree is the sum of the scores of all nodes in the tree:

$$s(x, y, \Theta) = \sum_{v \in y} \mathbf{v} \cdot \mathbf{h}_v^{(K)} \tag{4}$$

Training Given an input sentence x, the input to the reranker is the corresponding correct parse tree y and a list of trees generated by a base parser $\mathbf{gen}(x)$. As in conventional ranking systems, all discriminative rerankers can be trained with a margin-based hinge loss so that the score of the correct tree is higher than the score of the incorrect one with a margin of at least m:

$$L(y, t) = \max(0, s(x, t, \Theta) + m - s(x, y, \Theta)) \quad t \in \mathbf{gen}(x) \setminus \{y\} \tag{5}$$

Zhu et al. (2015) use a *structured margin* $m = \kappa\Delta(y, t)$, which is computed by counting the number of incorrect edges of t with respect to y. κ is a discount hyperparameter indicating the importance of Δ to the loss. In addition, the tree predicted by the model $\hat{y}$ (i.e., the highest scored tree) (3) is used to calculate the final loss. Alternatively, the loss of the predicted tree can be replaced by the average loss over all trees in the list.

3.3 Mixture reranking models

None of the models above does consider the scores from the base parser when ranking trees. Therefore, it seems plausible to try combining the advantages from both models, base parser and reranker, to produce a better final model. The most common way to do so is to consider the base parser and the reranker as a mixture model. The score of any reranking model s_r can be combined with the score of the base parser s_b using a linear combination:

$$s(x, y) = \alpha s_r(x, y, \Theta) + (1 - \alpha) s_b(x, y) \tag{6}$$

where $\alpha \in [0, 1]$ is a parameter.

4 Evaluating Neural Rerankers for Dependency Parsing

We are now providing a systematic evaluation of different neural reranking models used to rank the k-best lists generated by different parsers. In our first experiments, we try to reproduce the results for the available rerankers (IORNN, RCNN) on English. After that, we compare the performance of the rerankers on German and Czech data. Unless stated otherwise, results are compared based on UAS and LAS *including punctuation*.

4.1 Data

English Following Zhu et al. (2015), we use the Penn Treebank (PTB) with standard splits: sections 2-21 for training, section 22 for development and section 23 for testing. Their reranking models are applied to *unlabeled* trees. The authors used the linear incremental parser from Huang and Sagae (2010) to produce k-best lists and achieved slight improvements due to differences in optimization. In contrast, we obtained the data and pretrained model from the public repository.[1] Although not emphasized in their paper, Zhu et al. (2015) obtained the top k parses from the *forests* (a by-product of dynamic programming) rather than by using beam search. This is very important for reranking because the forest encodes exponentially many trees and so the k-best list extracted from the parse forest has a higher upper bound (Huang and Sagae, 2010).

Following previous work, we refer to the greedy, one-best results from the base parser as the *baseline*. *Oracle worst* and *best* are the lower and upper bound accuracies of the trees in the k-best list, respectively. *Top tree* results are calculated on the highest scored trees by the base parser in the list.

Table 1 shows that both our baseline and upper bound results are lower than those from Zhu et al. (2015). Extracting the top trees from the parse forest results in a much higher upper bound (+3.97%, development set) compared to using beam search (+1.46%, although not shown here). The maximum gain of our k-best list at $k = 64$ using the forest is about 1% lower than in Zhu et al. (2015).

German We use the German dataset from the SPMRL 2014 Shared Task (Seddah et al., 2014) which contains 50,000 sentences of newspaper text.

[1] https://github.com/lianghuang3/lineardpparser

	UAS w/ punct.		UAS w/o punct.	
Dataset	Dev	Test	Dev	Test
Zhu et al. (2015)				
Baseline	-	-	92.45	92.35
$k = 64$				
Oracle worst	-	-	73.30	-
Oracle best	-	-	97.34	-
Huang and Sagae (2010)				
Baseline	91.34	91.45	92.09	92.05
$k = 10$, forest				
Top tree	91.34	91.45	92.09	92.05
Oracle worst	79.68	79.56	80.21	80.19
Oracle best	95.31	95.33	95.99	95.82
$k = 64$, forest				
Top tree	91.34	91.45	92.09	92.05
Oracle worst	70.62	70.72	71.26	71.51
Oracle best	96.06	96.15	96.65	96.55

Table 1: Accuracy for k-best list from PTB. Top: accuracies reported in Zhu et al. (2015). Bottom: our k-best lists extracted with Huang and Sagae (2010)'s model using the parse forests.

We follow the original train/dev/test splits and use the predicted POS and morphological tags provided by the shared task organizers. The top k parses are produced using the graph-based parser in the MATE tools (Bohnet, 2010),[2] a non-neural model that employs second order, approximate non-projective parsing (McDonald and Pereira, 2006). The algorithm first finds the highest scored projective tree with exact inference, then rearranges the edges one at a time as long as the overall score improves and the parse tree does not violate the tree constraint. This algorithm also creates a list of k-best trees through its search process. We also tried to generate the k-best lists with a transition-based parser by adding a beam search decoder, but the beam failed to improve the parsing upper bound.

Czech We use the Czech Universal Dependencies (UD) Treebank,[3] based on the Prague Dependency Treebank 3.0 (Bejček et al., 2013). We use the original train/dev/test split and use MarMoT (Mueller et al., 2013) to predict UD POS tags by 5-way jackknifing. The k-best lists are created using the same parser as for German.

The properties of the k-best lists extracted from the German and Czech data are shown in table 2. Extracting the top k parses results in scores lower than the baseline when using the top trees as output, as the reranking scores do not always correlate with the quality of the trees.

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

[3] https://universaldependencies.org/

Dataset	Dev		Test	
	UAS	LAS	UAS	LAS
German				
Baseline	92.91	91.04	90.19	87.90
$k = 50$				
Top tree	91.75	90.04	88.36	86.28
Oracle worst	81.20	79.48	79.04	77.12
Oracle best	96.40	95.08	93.51	91.71
Czech				
Baseline	92.22	89.30	91.87	88.85
$k = 50$				
Top tree	91.02	88.28	90.74	87.93
Oracle worst	82.24	79.68	81.98	79.32
Oracle best	95.04	92.71	94.70	92.29

Table 2: k-best list accuracies for the German SPMRL and Czech UD datasets.

Pre-trained word embeddings In all experiments on English, we use the 50-dimensional GloVe word embeddings (Pennington et al., 2014) trained on Wikipedia 2014 and Gigaword 5. For German, we train 100-dimensional dependency-based word embeddings (Levy and Goldberg, 2014) on the SdeWaC corpus (Faaß and Eckart, 2013) with a cutoff frequency of 20 for both words and contexts and set the number of negative samples to 15. In experiments on Czech, we reduce the number of dimensions of the word vectors from fastText (Bojanowski et al., 2017) to 100 using PCA (Raunak et al., 2019).

4.2 Reproducing reranking results for PTB

This section is dedicated to the reproduction of the published results for the **IORNN** and **RCNN** rerankers on the English PTB. All results are from one run since we observe little variation between different runs[4] (and even between different settings the results hardly vary).

IORNN The results from Le and Zuidema (2014) can be reproduced with 93.01% UAS using the data and instructions from the public repository[5]. We are able to replicate this trend on our unlabeled English data described in §4.1, i.e., the reranking results are better than the baseline. The IORNN mixture model achieves 92.06% UAS on the test set, which is lower than the reproduced results on the paper's original data. Our baseline, however, is also lower due to the use of different data conversion rules for the conversion from constituency trees to dependencies, and the use of different base parsers. Note that Le and Zuidema (2014) also optimize the results on k while we keep k fixed in our experiments to make the results comparable between the different models. In addition, the authors do a logarithmic scaling for the score of the reranker in the mixture model combination (equation 6) and we use this function as it is.[6,7]

Model	UAS	LAS
Le and Zuidema (2014)		
Baseline	91.99	89.97
Oracle best ($k = 10$)	96.24	93.73
Reranker ($k = 6$)	92.83	90.76
Mixture ($k = 9$)	93.08	91.02
Reproduction on our data		
Baseline	91.45	-
Oracle best ($k = 10$)	95.33	-
Reranker ($k = 10$)	91.70	-
Mixture ($k = 10$)	92.06	-

Table 3: IORNN reranker results on the PTB test set

Table 3 summarizes the results from our reproduction study.

RCNN Since the code is not publicly available, we re-implemented the RCNN model following the description in the paper (Zhu et al., 2015). However, we were not able to reproduce the results on the 10-best list extracted from the parse forest. The authors report 93.83% (+1.48) UAS without punctuation using the mixture reranker with $k = 64$, and the same trend sets for all k. All our attempts to get better results than the base parser fail. Even when combining the reranking score with the score from the base parser, results do not improve over the baseline.

We run an ablation study to investigate the effect of different hyperparameters on the model's performance. We achieve best scores (UAS 90.65% and 90.29%) on both development and test set when removing L2 and structured margin and replacing

[4]For instance, the standard deviations of 5 runs on the development and test sets are $\sigma_{dev} = 0.05$, $\sigma_{test} = 0.07$ (%) when running the best GCN model setting on the English data.

[5]https://github.com/lephong/iornn-depparse

[6]The IORNN code does not output the reranking scores to train a mixture model separately.

[7]Applying a scaling to either score only affects the range of the combination parameter α, not the final results.

k_{train}	k_{eval}	UAS
10	10	91.50
64	10	91.86
10	64	90.82
64	64	91.62

Table 4: Accuracies of the RCNN-shared (+BiLSTMs) model on the PTB development set with regard to the size of the k-best list

the largest margin with the average margin. However, one thing we noted during training is that the learning curves indicate severe overfitting. In conclusion, despite our efforts we were not able to reproduce the RCNN results from Zhu et al. (2015).

RCNN-shared As the learning curves for the RCNN models show severe overfitting, we propose to simplify the original model. The original RCNN has a large number of parameters, due to its use of different weight matrices and vectors for the POS tags of the current head-child pair. In the simplified model, we replace those matrices $\mathbf{W}^{(h,c)}$ and vectors $\mathbf{v}^{(h,c)}$ with a shared matrix $\mathbf{W}$ and vector $\mathbf{v}$. Word embeddings and POS embeddings (randomly initialized) are concatenated as the input to the RCNN. Following common practice, we also test a model where we place several BiLSTM layers before the RCNNs to learn better representations from the input embeddings (+BiLSTMs). By switching from RCNN to the RCNN-shared model, we are now able to beat the baseline, even though by only a small margin (UAS 90.65% and 90.29% on the dev and test sets respectively).

We also study the effect of k to the model's performance (table 4). Training the reranker on a larger k-best list[8] improves the UAS by 0.36% on the development set, which shows that the model learns better with more negative examples. Increasing k at test time, on the other hand, hurts performance because the longer list now contains more low quality trees. The drop caused by using a longer list at test time is also smaller (0.20% vs 0.68%) when the model is trained with more trees.

4.3 Reranking with GCNs

We now present results for our new GCN reranking model on the English data. The best GCN model

Model	UAS
Baseline	91.45
IORNN ($k_{train} = 10$)	
Reranker ($k_{test} = 10$)	91.70
Mixture ($\alpha = 0.015$)	92.06
RCNN ($k_{train} = 10$)	
Reranker ($k_{test} = 10$)	90.29
Mixture ($\alpha = 0.005$)	91.53
With oracle	92.34
RCNN-shared (+BiLSTMs, $k_{train} = 64$)	
Reranker ($k_{test} = 10$)	91.75
Mixture ($\alpha = 0.05$)	91.92
With oracle	94.37
Reranker ($k_{test} = 64$)	91.43
Mixture ($\alpha = 0.01$)	92.21
With oracle	93.56
GCN ($k_{train} = 64$)	
Reranker ($k_{test} = 10$)	92.23
Mixture ($\alpha = 1.0$)	92.23
With oracle	95.25
Reranker ($k_{test} = 64$)	92.11
Mixture ($\alpha = 0.01$)	**92.48**
With oracle	94.69

Table 5: Results for different rerankers (PTB test set).

(using 1 BiLSTM layer and 3 GCN layers) trained on $k = 64$ parse trees significantly outperforms the RCNN-shared model[9] with 92.40% UAS on the development set, compared to 91.86% for RCNN-shared ($p < .001$), an increase of +0.54%.

The best results for the different reranking models on the PTB test set are summarized in table 5. We include in the table the results for reranking the top parse trees of different sizes ($k = 10, 64$). *Reranker* is the ranked list produced by the reranking model only. *Mixture* is the result for combining the output score given by the rerankers and the score of the base parser as described in §3.3. Following Zhu et al. (2015), we do not use the exact linear equation (6), but do logarithmic scaling of the base parser's score. The parameter α is optimized based on the results on the development set, which has the same k as the test set. Since the correct tree is not always in the k-best list, we also show an upper bound performance for our

[8] In practice, we do not train on the whole k-best trees when k is large, but down-sample k in each batch to keep the training time efficient. See the appendix for details.

[9] We did not do a hyperparameter optimization, but increased the number of parameters in the best RCNN-shared models and observed no significant improvement.

Model	UAS w/o punct.	Δ
Zhu et al. (2015)		
Baseline	92.35	
Mixture reranker	93.83	+1.48
With oracle	94.16	
Ours (Mixture GCNs)		
Baseline	92.05	
Mixture reranker	93.06	+1.01
With oracle	94.99	

Table 6: Reproduction of reranking results on the PTB test set for the GCN reranker ($k = 64$).

rerankers where we manually add the gold trees to the input list (*with oracle*). Note that *with oracle* is the result from the reranker, not from the mixture reranking model because the correct tree does not have a score from the base parser if it is not included in the k-best list.

Combining the score from both the reranker and the base parser consistently improves over the reranking score alone (except for the GCN reranker $k_{test} = 10$), which confirms our hypothesis that the parser and the reranker complement each other by looking at different scoring criteria. Although the accuracy drops when reranking longer lists, the mixture scores are always higher. Compared to the RCNN-shared models, the GCN models benefit less from the mixture models, maybe because the GCNs rank trees more similar to the base parser.

The upper bound performance (*with oracle*) shows that we can still improve results with a better k-best list. Interestingly, although we achieve modest improvements compared to Zhu et al. (2015), our upper bound is higher than theirs. A comparison of results with the original RCNN paper on their data is given in table 6.

4.4 Neural Reranking for MRLs

We now evaluate the reranking models that have proved to be effective for English (IORNN, RCNN-shared (+BiLSTMs) and GCNs) on German and Czech data. Note that the RCNN model only ranks *unlabeled* trees while the other two models also consider the dependency labels, which is particularly important for non-configurational languages. All models are trained with the same hyperparameter settings as for English. The mixture scores are combined using equation 6 except that we optimize the IORNN mixture model using the original tool provided by the authors.

The results for the different reranking models are presented in table 7 and 8. Neither the IORNN nor the RCNN-shared reranker can surpass the baseline. The GCN mixture model is the only model that shows significant improvements over the other models ($p < .001$) including the baseline, although small (∼0.15-0.3% LAS).

Taking a closer look at different grammatical functions in the output, we can see a clear differ-

Model	UAS	LAS
Baseline	90.19	87.90
Top tree	88.36	86.28
IORNN ($k_{train} = 10$)		
Reranker ($k_{test} = 10$)	89.32	87.16
Mixture ($\alpha = 0.91$)	89.47	87.41
RCNN-shared ($k_{train} = 50$)		
Reranker ($k_{test} = 50$)	89.50	86.12
Mixture ($\alpha = 0.1$)	90.12	87.87
With oracle	92.76	90.06
GCN ($k_{train} = 50$)		
Reranker ($k_{test} = 50$)	89.96	87.50
Mixture ($\alpha = 0.11$)	**90.33**	**88.21**
With oracle	94.29	92.85

Table 7: Performance of different rerankers on the German SPMRL test set.

Model	UAS	LAS
Baseline	91.87	88.85
Top tree	91.02	88.28
IORNN ($k_{train} = 10$)		
Reranker ($k_{test} = 10$)	91.07	87.97
Mixture ($\alpha = 0.94$)	91.42	88.54
RCNN-shared ($k_{train} = 50$)		
Reranker ($k_{test} = 50$)	90.68	86.63
Mixture ($\alpha = 0.07$)	91.79	88.80
With oracle	93.28	89.99
GCN ($k_{train} = 50$)		
Reranker ($k_{test} = 50$)	91.12	87.84
Mixture ($\alpha = 0.09$)	**91.89**	**89.01**
With oracle	94.47	92.42

Table 8: Performance of different rerankers on the Czech UD test set.

	German		Czech	
Label	Baseline	Δ_{GCNs}	Baseline	Δ_{GCNs}
nsubj	89.20	1.48	91.50	0.46
obj	82.84	1.91	90.10	0.54
iobj	67.25	1.15	60.92	2.22
conj	81.78	0.72	74.15	1.23

Table 9: Labeled F1 differences between the baseline and the GCN mixture model for selected dependency types from the German and Czech test sets.

ence between the reranking results and the baseline (table 9). Although the overall accuracy is similar, our reranking results show a better performance for core arguments (nsubj: subject, obj: direct object, iobj: indirect object) and conjunctions (conj).

4.5 Analysis

Through our experiments, we have shown that neural reranking models, which have demonstrated their effectiveness on English data, fail to improve baseline parsing results when applied to German and Czech. This brings us to the question whether this failure is due to the differences between the languages or simply due to the lower quality in the German and Czech k-best lists that are input to the rerankers. It is conceivable that language-specific properties such as the freer word order and richer morphology in German and Czech might make it harder for our models to learn a good representation capturing the quality of a specific parse tree. However, when we add the correct parse tree to the k-best list (*with oracle* results in table 5, 7 and 8), the accuracy goes up to 94% for English, German and Czech, which effectively eliminates the first reason.

This points to the method used to obtain the k-best list as the main factor responsible for the low results for German and Czech. Beam search, although being straightforward to implement, fails to create high quality k-best lists for the base parsers used for both languages (§4.1). While several projective parsers support k-best parsing (Huang and Sagae, 2010; McDonald and Pereira, 2006), there is, to the best of our knowledge, no out-of-the-box parsing system that implements an effective non-projective k-best parsing algorithm (as, for example, Hall (2007)'s algorithm).

Gold tree ratio Clearly, the (upper bound) tree accuracy in the k-best list determines the reranking performance. In all datasets, we observe that the accuracy decreases when sentence length increases. Overall, the (unlabeled) tree accuracy in the English k-best list is ∼5% higher than in the German data, but is behind that in the Czech data. This, however, is not caused by a larger amount of long sentences in the German data. For sentences of same length, the top k trees from the PTB contain more gold trees than those from the German SPMRL and Czech UD datasets.

We further study the effect of the gold tree ratio for reranking by removing the gold trees from the k-best list to reduce the ratio to a certain level. Figure 1 shows that the gold tree ratio strongly correlates with the reranking results.

k-best list variation We measure the variation between the trees in the k-best lists by calculating the standard deviation of their UAS. Figure 2 illustrates the UAS standard deviation distribution in the data for the three languages for $k = 10$. In each dataset, the tree UAS variation in the English data is the highest, followed by German and then Czech, which shows that the re-arranging method used to generate German and Czech k-best trees tends to return more similar trees. We hypothesize

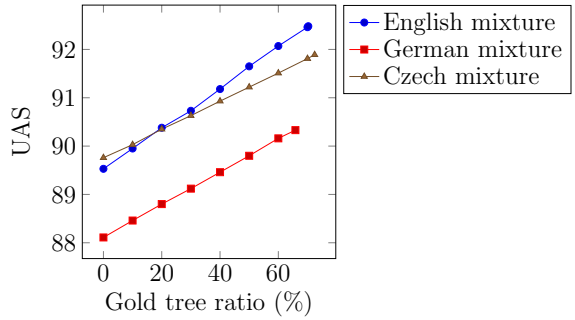

Figure 1: UAS for the GCN reranking mixture model with respect to the gold tree ratio in the k-best lists.

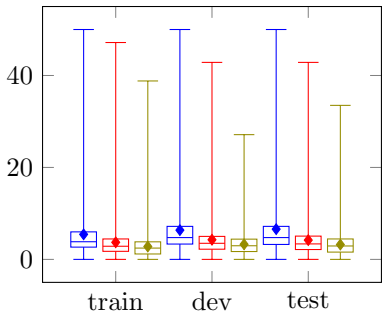

Figure 2: Tree UAS standard deviation of 10-best lists. From left to right: English, German, Czech.

that reranking benefits from diversity, especially if the data contains hard negative examples (incorrect trees that are very similar to the correct one). The gap between reranker performance and *with oracle* results shows that the reranker is able to detect the correct tree among the incorrect ones because they are very different from each other.

Reranking models Among the neural rerankers, the RCNNs are prone to error propagation from the lower levels, and the IORNNs are sensitive to the order of the child nodes. Both models did not work very well when moving to German and Czech compared to the GCNs, which disregard the top-down or left-to-right order.

In practice, parser output reranking is not a very cost effective way to improve parsing performance, unless we have a fast way to generate high quality output trees. However, the small improvement in core arguments might be useful for downstream applications that require high quality prediction of core arguments.

5 Conclusion

We have evaluated recent neural techniques for reranking dependency parser output for English, German and Czech and presented a novel reranking model, based on graph convolutional networks (GCNs). We were able to reproduce results for English, using existing rerankers, and showed that our novel GCN-based reranker even outperformed them. However, none of the rerankers works well on the two morphologically rich(er) languages.

Our analysis gave some insights into this issue. We showed that the failure of the rerankers to improve results for German and Czech over the baseline is due to the lower quality of the k-best lists. Here the gold tree ratio in the k-best list plays an important role, as the discriminative rerankers are very well able to distinguish the gold trees from other trees in the list, but their performance drops notably when we remove the gold trees from the list. In addition, we observe a higher diversity in the English k-best list, as compared to German and Czech, which helps the rerankers to learn the differences between high- and low-quality trees.

We conclude that the prerequisite for improving dependency parsing with neural reranking is a diverse k-best list with a high gold-tree ratio. The latter is much harder to achieve for MRLs where the freer word order and high amount of non-projectivity result in a larger number of tree candidates, reflected by a lower gold tree ratio.

Acknowledgments

This work was supported by 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.

References

Eduard Bejček, Eva Hajičová, Jan Hajič, Pavlína Jínová, Václava Kettnerová, Veronika Kolářová, Marie Mikulová, Jiří Mírovský, Anna Nedoluzhko, Jarmila Panevová, Lucie Poláková, Magda Ševčíková, Jan Štěpánek, and Šárka Zikánová. 2013. Prague dependency treebank 3.0. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Bernd Bohnet. 2010. Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 89–97, Beijing, China. Coling 2010 Organizing Committee.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 173–180, Ann Arbor, Michigan. Association for Computational Linguistics.

Michael Collins and Terry Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–70.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *The 5th International Conference on Learning Representations*, Toulon, France.

Gertrud Faaß and Kerstin Eckart. 2013. SdeWaC - A corpus of parsable sentences from the web. In *Language Processing and Knowledge in the Web: 25th International Conference, GSCL 2013, Darmstadt, Germany, September 25-27, 2013. Proceedings*, pages 61–68, Berlin, Heidelberg. Springer.

Keith Hall. 2007. K-best spanning tree parsing. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 392–399, Prague, Czech Republic. Association for Computational Linguistics.

Katsuhiko Hayashi, Shuhei Kondo, and Yuji Matsumoto. 2013. Efficient stacked dependency parsing by forest reranking. *Transactions of the Association for Computational Linguistics*, 1:139–150.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of ACL-08: HLT*, pages 586–594, Columbus, Ohio. Association for Computational Linguistics.

Liang Huang and David Chiang. 2005. Better k-best parsing. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 53–64, Vancouver, British Columbia. Association for Computational Linguistics.

Liang Huang and Kenji Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1077–1086, Uppsala, Sweden. Association for Computational Linguistics.

Phong Le and Willem Zuidema. 2014. The inside-outside recursive neural network model for dependency parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 729–739, Doha, Qatar. Association for Computational Linguistics.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308, Baltimore, Maryland. Association for Computational Linguistics.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1506–1515, Copenhagen, Denmark. Association for Computational Linguistics.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *11th Conference of the European Chapter of the Association for Computational Linguistics*.

Thomas Mueller, Helmut Schmid, and Hinrich Schütze. 2013. Efficient higher-order CRFs for morphological tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 322–332, Seattle, Washington, USA. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Vikas Raunak, Vivek Gupta, and Florian Metze. 2019. Effective dimensionality reduction for word embeddings. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 235–243, Florence, Italy. Association for Computational Linguistics.

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*, pages 103–109, Dublin, Ireland. Dublin City University.

Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. 2013. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 455–465, Sofia, Bulgaria. Association for Computational Linguistics.

Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2010. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS 2010 Deep Learning and Unsupervised Feature Learning Workshop*.

Hao Zhou, Yue Zhang, Shujian Huang, Junsheng Zhou, Xin-Yu Dai, and Jiajun Chen. 2016. A search-based dynamic reranking model for dependency parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1393–1402, Berlin, Germany. Association for Computational Linguistics.

Chenxi Zhu, Xipeng Qiu, Xinchi Chen, and Xuanjing Huang. 2015. A re-ranking model for dependency parser with recursive convolutional neural network. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1159–1168, Beijing, China. Association for Computational Linguistics.

A Appendix: Training Details for Neural Network Rerankers

A.1 Down-sampling the k-best list

In order to maintain an efficient run-time for our discriminative rerankers without scarifying the diversity we get from a longer k-best list, we apply down-sampling for each training instance. Namely, in each step, we use only 10 randomly selected trees (when $k > 10$) in addition to the gold tree for each sentence to back-propagate.

A.2 IORNN reranker

We use the code provided by Le and Zuidema (2014)[10] to train all IORNN rerankers with default hyperparameters for English, German and Czech. The default number of training epochs is set to 50. Due to time limit, we could only train the model for Czech which is the largest of our datasets up to 27 epochs, which took 15 days on a CPU. The program processes a single sentence at a time rather than batching or multithreading. For computing the mixture score, we use the tool provided in the repository instead of ours. The authors do logarithmic scaling for the score of the reranker in the mixture model combination:

$$s(x, y) = \alpha \log s_r(x, y, \Theta) + (1 - \alpha) s_b(x, y)$$

A.3 RCNN, RCNN-shared and GCN rerankers

For all discriminative rerankers, in the experiment with the English data, we do logarithmic scaling for the score of the base parser in the mixture model combination:

$$s(x, y) = \alpha s_r(x, y, \Theta) + (1 - \alpha) \log s_b(x, y)$$

In the experiments with German and Czech data, we do not scale the score of the base parser and use equation 6.

[10] https://github.com/lephong/iornn-depparse