On Interpretable Reranking-Based Dependency Parsing Systems

Florian Schottmann[1], Vincent Fortuin[1], Edoardo Ponti[2] and Ryan Cotterell[1]

[1]*Department of Computer Science, ETH Zurich, Zurich, Switzerland*
[2]*Mila AI Centre, Quebec, Canada*

Abstract
Reranking the k best hypothesis parse trees from an existing parser allows to take into account more information that the model has gathered during training than simply decoding the most likely dependency tree. In this paper, we first investigate whether state-of-the-art dependency parsers can still benefit from reranking in low-resource languages. As part of this analysis, we deliver new insights concerning rerankability. Second, we propose a reranker reject option, which paves the way for designing interpretable reranking-based parsing systems in the future.

1. Introduction

Dependency parse trees model the structural relationships in sentences [1]. They are relevant for many downstream applications like natural language understanding tasks [2] or information extraction [3].

State-of-the-art dependency parsers predict the most likely dependency parse tree for a given sentence based on large neural network architectures [4]. By acting greedily with respect to the prediction, these models disregard information that was gathered during training. In particular, the best dependency tree might not have been assigned the highest probability, for example due to unusual training sentences [5]. A technique that tries to remedy this problem is k-best reranking. It is based on decoding the k best predictions from the base parser and reranking these using an additional machine learning model [6].

In this paper, we are investigating whether interpretable rerankers can improve current state-of-the-art dependency parsers. We challenge previous conclusions in which situations base parsers can be reranked and propose a novel approach to combining the base parser and the reranker.

2. Background

Many state-of-the-art dependency parsing systems are graph-based parsers, which are based on an edge-factored model (e.g. 4 and 7). The underlying assumption of an edge-factored model is that the score of a dependency tree can be factored across the edges of the graph [1]. In this work, we define the set of edges of a dependency tree T as E_T. The edges consist of a relation label $r \in R$ (where R is the set of all possible relation labels) and two tokens w_i and w_j, i.e. $(w_i, r, w_j) \in E_T$. If we use the conditional probability of a dependency tree (given a particular sentence S) as a scoring function, we can write the model as

$$p(T|S;\boldsymbol{\theta}) = \frac{1}{Z} \prod_{(w_i,r,w_j)\in E_T} \exp\{s(w_i, r, w_j; \boldsymbol{\theta})\} \tag{1}$$

$$Z = \sum_{T' \in \mathcal{T}_{\text{full}}(S)} \prod_{(w_i,r,w_j)\in E_{T'}} \exp\{s(w_i, r, w_j; \boldsymbol{\theta})\}$$

where $\mathcal{T}_{\text{full}}(S)$ is the set of all dependency trees for sentence S. s is the edge scoring function and $\boldsymbol{\theta}$ is the parameter vector.

From the model definition, we can see that any edge-factored model learns $n \times n$ edge scores for a sentence S, where n is the number of tokens in the sentence augmented by an artificial *root* token at the beginning of the sentence. Decoding the k best dependency trees from such a score matrix after training is not possible naively, since the search space is exponential [8].

Recently, Zmigrod et al. [9] provided an algorithm which allows to decode the k best dependency trees from an adjacency matrix induced by an edge-factored model, which we will use in this work to construct the list of hypothesis trees.

Often, building a reranking model on top of the base parser is not yet enough for state-of-the-art performance. A popular strategy to improve a reranking model is mixture reranking (MR), i.e. combining the scores of the base parser and the reranker with a trade-off parameter that is tuned on the development set [10, 11]:

$$s_f = \alpha\, s_{RR}(T, \boldsymbol{\theta}_{RR}) + (1 - \alpha)\, s_B(T, \boldsymbol{\theta}_B) \tag{2}$$

where $\alpha \in [0, 1]$ is the mixing parameter. s_{RR} is the score of the reranker and s_B is the score of the base parser given

SwissText 2022: Swiss Text Analytics Conference, June 08–10, 2022, Lugano, Switzerland
fschottmann@ethz.ch (F. Schottmann);
fortuin@inf.ethz.ch (V. Fortuin);
edoardo-maria.ponti@mila.quebec (E. Ponti);
ryan.cotterell@inf.ethz.ch (R. Cotterell)

a tree T. $\boldsymbol{\theta}$ are the respective parameter vectors and s_f is the final score of the tree T.

3. K-best list

Recently, Do and Rehbein [5] emphasized that, apart from the oracle parsing accuracy [6], the quality of the k-best lists of a dataset can be indicated by the gold tree ratio and the unlabeled attachment scores (UAS) standard deviation in the k-best list. The gold tree ratio refers to the number of times a correct tree is in the k-best list relative to the number of sentences in the respective dataset. The oracle parsing accuracy is a similar metric introduced by Hall [6], referring to the maximum score possible by always choosing the closest tree to the gold tree in the k-best list.

4. Reranker reject option

In this work, we reframe mixture reranking by using a reject option for the reranker. A reject option is a concept from decision theory; it refers to the idea of rejecting a classification decision if the associated probability is lower than a certain threshold τ [12]. In the context of reranking, the reject option works as follows:

$$T^* = \begin{cases} \underset{T' \in \mathcal{T}_k}{\operatorname{argmax}}\ p(T'|S) & \text{if } \max_{T' \in \mathcal{T}_k} p(T'|S) > \tau \\ T_1 & \text{else} \end{cases} \tag{3}$$

where $\mathcal{T}_k$ is the set of candidate parses for sentence S and τ is the confidence threshold. If the confidence of the reranker is less than or equal to τ, the prediction of the base parser T_1 is used.

Compared to mixture reranking, our method offers a more interpretable way of trading off reranker and base parser. When using mixture reranking, it is not clear how much relative weight the reranker and the base parser get in the final decision of a parsing system if the base parser's score is not a probability (unless $\alpha \in \{0, 1\}$). Indeed, the base parser score of Qi et al. [7] is not normalized over all valid dependency trees. By using a reranker reject option, we do not rely on the score of the base parser: We tune a threshold of minimum certainty that implicitly trades off reranker and base parser, but always leads to a clear decision whether the reranker or the base parser takes the final decision of the parsing system.

5. Experimental Setup

5.1. Training

We ran our experiments on four low-resource languages using data from the Universal Dependencies v2.5 treebanks [13], since our chosen base parser's performance offers most potential for improvement in the low-resource domain [14]. In particular, we decided to use Lithuanian, Belarussian, Marathi, and Tamil. The data split is indicated in Table 1.

Table 1
Dataset Lengths

Treebank	Train	Dev	Test
Lithuanian HSE	153	55	55
Belarussian HSE	319	65	253
Marathi UFAL	373	46	47
Tamil TTB	400	80	120

We use the base parser of Qi et al. [7] and decode the k-best list by using the algorithm of Zmigrod et al. [9]. As reranking models, we train structured support vector machine (SVM) and Gaussian process (GP) classification models on each of the four languages. All models are kernelized with the kernel of Collins et al. [15], which measures the similarity of two dependency trees in terms of the number of subtrees that they have in common. Thus, the kernel takes structural overlap as a measure of similarity between the trees [15]. Except for the kernel, no other features are used. We refrain from using neural-network-based rerankers to avoid the construction of black-box features.

We fix the random seeds to guarantee reproducibility of our models. The implementation of the GPs (in particular the computation of the GP posterior) does not allow full reproducibility of the fitting procedure. To account for this randomness, we train the GP models over five different seeds.

We tune the inverse regularization parameter c (only in case of the SVM), k (the number of candidate trees at prediction time) and α or τ simultaneously on the development set. In case of ties, we take the parameter combination corresponding to maximum regularization with the highest number of candidates. That corresponds to making a conservative decision while maximizing the diversity of the candidate parses, where the latter has been suggested to be useful by Do and Rehbein [5]. We choose the hyperparameter combination that achieves the highest (average) labeled attachment score (LAS) on the development set.

5.2. Evaluation

The baseline performance is generated by feeding the pre-tokenized sentences into the base parser and evaluating the predictions with the official CoNLL 2018 shared task evaluation script. That leads to considerably higher LAS scores than reported by Qi et al. [14]. We hypothesize that this is the result of the fixed tokenization and the fixed sentence split, given that the CoNLL 2018 shared

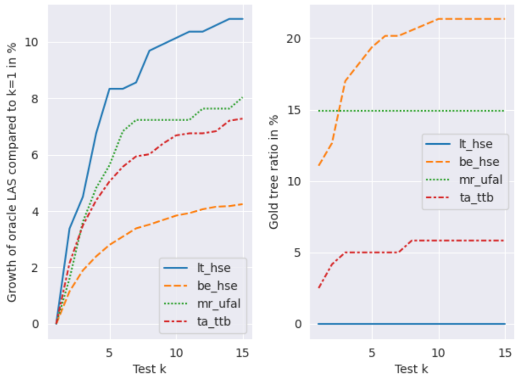

Figure 1: Growth of oracle LAS and gold tree ratio

task required a prediction from raw text.

All trained models are evaluated in terms of the labeled attachment score (LAS), which measures how many words were assigned the correct head with the correct dependency label. In particular, it is an F1 score, i.e. the harmonic mean of labeled precision and labeled recall [16].

6. Results

6.1. K-best list

Figure 1 shows the relative growth of the oracle LAS for different k. All values are with respect to the base parser score, which explains why the curve is weakly increasing for every language. The growth for Lithuanian is the largest with more than 10%, whereas the oracle LAS grows less than 5% for Belarussian. Marathi and Tamil are in between with a growth of approximately 8%.

The gold tree ratio is also weakly increasing, given that more candidate trees can never lead to fewer gold trees in the set of candidate parses. The gold tree ratio is constantly at zero for Lithuanian. For Tamil, it is always near 5%, while for Marathi it is stable at 15%. The only gold tree ratio that shows significant improvement is for Belarussian, from approximately 10% to approximately 20%.

6.2. Reranking performance

The performance in terms of the LAS of the different parsing systems is indicated in Table 2. We can see that augmenting the base parser by an interpretable reranker does not improve parsing performance drastically.

Looking at the chosen hyperparameters in Table 3, we can see that both methods are highly sensitive to the reranking model.

For all investigated languages, the SVM-based system with the reranker reject option trusts the reranker. This is indicated by $\tau = 0$. The corresponding system with mixture reranking reproduces this decision for Lithuanian and Marathi (i.e. $\alpha = 1$), while it involves the base parser in the classification decision for Belarussian and Tamil. Still, α is also close to 1 in these cases, indicating the proximity to the reranker reject option. Interestingly, we can see in Belarussian and Tamil that the system with the reranker reject option is more regularized (i.e. smaller optimal c and k) than the system with mixture reranking (which involves the base parser). Both SVM-based systems slightly outperform the base parser in Lithuanian.

The GP-based systems trust the reranker less. For Tamil, both systems fully commit to the base parser (i.e. $\alpha = 0$ and $\tau = 1$). Note that in these situations, based on the principles by which we choose the hyperparameters (see Section 5.1), the chosen k always equals 15. Both GP-based systems take related decisions for Marathi, where both involve base parser and reranker in the final decision. This leads to a slight average outperformance. Interestingly, the system with the reranker reject option generally chooses a smaller optimal k (also for the SVM-based systems). The decisions of both GP-based systems are almost contrary for Lithuanian and Belarussian. None is clearly better than the other based on the results.

7. Discussion

Based on Figure 1, we generally see that even the strong base parser of Qi et al. [7] can benefit from k-best reranking. We hypothesize that the gains in oracle LAS of additional candidates are crucial for reranking since those were the highest for Lithuanian and Marathi (the only languages where the base parser could be reranked on average).

Do and Rehbein [5] state that a prerequisite for successful neural reranking is a high gold tree ratio. In our opinion, this is not necessarily the case, given that we were able to realize small improvements on a language without a single gold tree in the k-best lists (i.e. Lithuanian) with non-neural models. In Belarussian, the language with the highest gold tree ratio, on the other hand, the reranker did not bring any benefit. The reason why Belarussian is hard to rerank might be that the base parser's predictions are on average already close to the true trees. This is indicated by a steeply rising gold tree ratio but only marginal gains in oracle LAS scores. Additional similar trees result in a high variance of the reranker. As a result, the reranker uses strong regularization. Still, the reranker performs worse than the base parser, supposedly because the kernel function is not ideal for capturing the fine-grained differences between these highly similar candidate trees [15]. In the

Table 2
LAS of different reranking-based parsing systems

	Lithuanian HSE			Belarussian HSE			Marathi UFAL			Tamil TTB		
	Mean	Median	Std	Mean	Median	Std	Mean	Median	Std	Mean	Median	Std
Base Parser	42.08	42.08	0	**81.55**	**81.55**	0	**60.44**	**60.44**	0	**67.67**	**67.67**	0
SVM MR	**42.17**	**42.17**	0	**81.55**	**81.55**	0	59.95	59.95	0	67.57	67.57	0
SVM Reject	**42.17**	**42.17**	0	81.53	81.53	0	59.95	59.95	0	**67.67**	**67.67**	0
GP MR	42.08	42.08	0	**81.56**	**81.55**	0.04	**60.49**	**60.44**	0.11	**67.67**	**67.67**	0
GP Reject	41.87	41.98	0.29	**81.55**	**81.55**	0	**60.53**	**60.68**	0.22	**67.67**	**67.67**	0

Table 3
Optimized hyperparameters of different reranking-based parsing systems

	Lithuanian HSE	Belarussian HSE	Marathi UFAL	Tamil TTB
SVM MR	$\alpha = 1,\ k = 6,\ c = 199$	$\alpha = 0.89,\ k = 15,\ c = 199$	$\alpha = 1,\ k = 2,\ c = 199$	$\alpha = 0.95,\ k = 15,\ c = 199$
SVM Reject	$\tau = 0,\ k = 6,\ c = 199$	$\tau = 0,\ k = 8,\ c = 0.003$	$\tau = 0,\ k = 2,\ c = 199$	$\tau = 0,\ k = 2,\ c = 0.003$
GP MR	$\alpha = 0,\ k = 15$	$\alpha = 0.89,\ k = 7$	$\alpha = 0.95,\ k = 15$	$\alpha = 0,\ k = 15$
GP Reject	$\tau = 0.3,\ k = 3$	$\tau = 1,\ k = 15$	$\tau = 0.3,\ k = 2$	$\tau = 1,\ k = 15$

case of Tamil, we see a similar pattern. The higher oracle LAS growth and the less steep increase in the gold tree ratio however lead to less similar candidate trees. In this situation, a strongly regularized reranker can reproduce the base parser performance, whereas biasing the reranker towards the base parser seems to rather distort the predictions.

Based on the performance in Table 2 and the hyperparameter selection in Table 3, we see that both mixing methods choose similar hyperparameters and perform almost the same. Generally, the reranker reject option works better if the classifier outputs well-calibrated probabilities. Tuning the hyperparameter τ might thus be harder than tuning α, since it is dependent on the quality of the reranker's predictions.

8. Conclusion

All in all, we can observe from Table 2 that the gains of interpretable rerankers (with the kernel of 15) are not enough to justify their computational cost. This is in line with the recent literature on k-best reranking [5]. Our experiments with the k-best list show a positive growth of the oracle LAS, meaning that k-best reranking provides possibilities to improve on strong base parsers. However, the tested models cannot consistently leverage this potential.

By proposing a reranker reject option, we pave the way for fully interpretable parsing systems. Similar in performance to mixture reranking [10, 11], it allows a more fine-grained analysis of rerankers and base parsers, since every parsing decision can be traced back to either of the parsing system's components. Future work towards interpretable reranking could involve more sophisticated kernel functions that are able to better measure the subtle differences between the candidate trees.

Acknowledgements

We thank Ran Zmigrod for insightful discussions at an early stage of the project.

References

[1] S. Kübler, R. McDonald, J. Nivre, Dependency parsing, Synthesis lectures on human language technologies 1 (2009) 1–127.

[2] S. R. Bowman, J. Gauthier, A. Rastogi, R. Gupta, C. D. Manning, C. Potts, A fast unified model for parsing and sentence understanding, in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Berlin, Germany, 2016, pp. 1466–1477. URL: https://aclanthology.org/P16-1139. doi:10.18653/v1/P16-1139.

[3] G. Angeli, M. J. Johnson Premkumar, C. D. Manning, Leveraging linguistic structure for open domain information extraction, 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), Association for Computational Linguistics, Beijing, China, 2015, pp. 344–354. URL: https://aclanthology.org/P15-1034. doi:10.3115/v1/P15-1034.

[4] M. Straka, J. Straková, J. Hajič, Evaluating contextualized embeddings on 54 languages in pos tagging, lemmatization and dependency parsing, arXiv preprint arXiv:1908.07448 (2019). URL: https://arxiv.org/pdf/1908.07448.pdf.

[5] B.-N. Do, I. Rehbein, Neural reranking for dependency parsing: An evaluation, in:

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Online, 2020, pp. 4123–4133. URL: https://aclanthology.org/2020.acl-main.379. doi:10.18653/v1/2020.acl-main.379.

[6] K. Hall, K-best spanning tree parsing, in: Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, Association for Computational Linguistics, Prague, Czech Republic, 2007, pp. 392–399. URL: https://aclanthology.org/P07-1050.

[7] P. Qi, T. Dozat, Y. Zhang, C. D. Manning, Universal Dependency parsing from scratch, in: Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, Association for Computational Linguistics, Brussels, Belgium, 2018, pp. 160–170. URL: https://aclanthology.org/K18-2016. doi:10.18653/v1/K18-2016.

[8] R. McDonald, F. Pereira, K. Ribarov, J. Hajič, Non-projective dependency parsing using spanning tree algorithms, in: Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Vancouver, British Columbia, Canada, 2005, pp. 523–530. URL: https://aclanthology.org/H05-1066.

[9] R. Zmigrod, T. Vieira, R. Cotterell, On finding the k-best non-projective dependency trees, in: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), Association for Computational Linguistics, Online, 2021, pp. 1324–1337. URL: https://aclanthology.org/2021.acl-long.106. doi:10.18653/v1/2021.acl-long.106.

[10] K. Hayashi, S. Kondo, Y. Matsumoto, Efficient stacked dependency parsing by forest reranking, Transactions of the Association for Computational Linguistics 1 (2013) 139–150. URL: https://aclanthology.org/Q13-1012. doi:10.1162/tacl_a_00216.

[11] Q. Le, T. Mikolov, Distributed representations of sentences and documents, in: E. P. Xing, T. Jebara (Eds.), Proceedings of the 31st International Conference on Machine Learning, volume 32 of *Proceedings of Machine Learning Research*, PMLR, Bejing, China, 2014, pp. 1188–1196. URL: http://proceedings.mlr.press/v32/le14.pdf.

[12] C. M. Bishop, Pattern Recognition and Machine Learning (Information Science and Statistics), Springer-Verlag, Berlin, Heidelberg, 2006.

[13] D. Zeman, J. Nivre, M. Abrams, E. Ackermann, N. Aepli, H. Aghaei, Universal dependencies 2.5, 2019. URL: http://hdl.handle.net/11234/1-3105, LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

[14] P. Qi, Y. Zhang, Y. Zhang, J. Bolton, C. D. Manning, Stanza: A Python natural language processing toolkit for many human languages, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations, 2020. URL: https://aclanthology.org/2020.acl-demos.14.pdf.

[15] M. Collins, N. Duffy, F. Park, Parsing with a single neuron: Convolution kernels for natural language problems, Technical report UCSC-CRL-01-01, University of California at Santa Cruz (2001). URL: http://luthuli.cs.uiuc.edu/~daf/courses/learning/Kernelpapers/collins01parsing.pdf.

[16] D. Zeman, J. Hajič, M. Popel, M. Potthast, M. Straka, F. Ginter, J. Nivre, S. Petrov, CoNLL 2018 shared task: Multilingual parsing from raw text to Universal Dependencies, in: Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, Association for Computational Linguistics, Brussels, Belgium, 2018, pp. 1–21. URL: https://aclanthology.org/K18-2001. doi:10.18653/v1/K18-2001.

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830. URL: https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf.

[18] GPy, GPy: A gaussian process framework in python, 2012. URL: http://github.com/SheffieldML/GPy.

A. Appendix

A.1. Implementation Details

We construct the training set for each of the chosen languages by decoding up to 9 trees from the weighted graph learned by the base parser with the algorithm of Zmigrod et al. [9]. Next, we add the corresponding true parse tree to the list. The k is viewed as a hyperparameter and is thus not fixed a priori.

We used the implementation of the SVMs from sklearn [17] and the GP implementation of GPy [18]. We set the hyperparameter λ of the Collins et al. [15] kernel to 0.7 based on preliminary experiments on the Lithuanian training set.

The hyperparameters c, k, α and τ are tuned simultaneously on the development set. The search space for the inverse regularization parameter c is $(0.0001, 0.0027, 0.7356, 199.5262)$.

The search space for k is $0 < k < 16$ for SVMs and $1 < k < 16$ for GPs. The latter is motivated by empirical reasons, i.e. the tendency of the GPs to always stick to the base parser unless they are forced not to. With both methods, the models can always fall back to the base parser if that is considered optimal.

For the search space for α, we generated 20 evenly spaced values in the interval $0 \leq x \leq 1$.

Finally, the search space for τ is $(0, 0.3, 0.7, 1)$. Note that none of the values is too close to 0.5, since we expect high variance in the results if values near 0.5 are used as a cutoff.