High-Accuracy Transition-Based Constituency Parsing

John Bauer
HAI
Stanford University
horatio@cs.stanford.edu

Christopher D. Manning
Linguistics & Computer Science
Stanford University
manning@stanford.edu

Abstract

Constituency parsers have improved markedly in recent years, with the F1 accuracy on the venerable Penn Treebank reaching 96.47, half of the error rate of the first transformer model in 2017. However, while dependency parsing frequently uses transition-based parsers, it is unclear whether transition-based parsing can still provide state-of-the-art results for constituency parsing. Despite promising work by Liu and Zhang in 2017 using an in-order transition-based parser, recent work uses other methods, mainly CKY charts built over LLM encoders. Starting from previous work, we implement self-training and a dynamic oracle to make a transition-based constituency parser, and test it on seven languages. Using Electra embeddings as the input layer on Penn Treebank, with a self-training dataset built from Wikipedia, our parser achieves a new SOTA F1 of 96.61.

1 Introduction

This work examines whether it is still possible to build a state of the art constituency parser using transition-based parsing.

Recent years have seen strong progress in both dependency and constituency parsers. Models for both of these tasks have progressed from using categorical features in PCFG models or shift-reduce parsers, to word embeddings, to transformers. Upgrading to a transformer can have a dramatic effect on the quality of the results (Nguyen et al., 2021; Vaswani et al., 2017).

While transition-based parsing is a widely used technique for dependencies, in recent years it has been less common for constituency parsing. There are still models built on transitions which are quite successful, such as Yang and Deng (2020), which uses an attention mechanism to predict how and where to attach the next word of a sentence. However, most state of the art models use a derivative of the CKY algorithm (Younger, 1967). For example, the current state of the art uses bidirectional attention as part of the encoder for a chart parser (Kim et al., 2023).

In this work, we revisit older transition schemes and show that they can be used to build a state-of-the-art model. We improve on existing dynamic oracle methods (Fernández-González and Gómez-Rodríguez, 2018), allowing the model to better learn the state space after an error. Furthermore, we explore self-training using a method inspired by active learning (Swayamdipta et al., 2020; Karamcheti et al., 2021), and present a simple way to ensemble transition models with the same structure. We present state of the art results on 6 languages. Using Electra embeddings with the Penn Treebank, our parser achieves a new SOTA F1 of 96.61.

All models and code described are publicly released in the (anonymous) software package.

2 Experimental Setup

We experiment on Penn Treebank 3 for English (Marcus et al., 1999) and the Chinese Treebank 5.1 for Simplified Chinese (Xue et al., 2005). We also report results for German, Indonesian, Italian, Japanese, and Vietnamese (Brants et al., 2001; Suan Lim et al., 2023; Delmonte et al., 2007; Thu et al., 2016; Ha et al., 2022). See table 1 for details.

See Appendix B for the hyperparameters used when training the model, and Appendix C for a more complete description of the datasets used.

Scores are reported as averages of 5 models.

Lang	Dataset	|Train|	|Dev|	|Test|
EN	PTB 3	39832	1700	2416
ZH	CTB 5.1	45893	2040	2725
DE	Tiger	40472	5000	5000
ID	ICON	8000	1000	1000
IT	VIT	7875	683	1042
JA	ALT	17195	934	931
VI	VLSP 22	8160	n/a	1204

Table 1: Datasets used in this work

Proceedings of the 18th International Conference on Parsing Technologies (IWPT, SyntaxFest 2025), pages 26–39
August 26, 2025 ©2025 Association for Computational Linguistics

3 Model Improvements

Improvements to Base Model. The parser builds from a base of the LSTM in-order transition-based parser of Liu and Zhang (2017), which itself builds from Dyer et al. (2015). Several specific improvements to this model improved scores slightly.

The first is that the original LSTM between the word embeddings and the classification layers was a unidirectional LSTM but using a bi-LSTM instead slightly improves performance. As we use self-training to boost performance, rather than reranking, we do not need an autoregressive parser.

Another minor improvement is to follow Bauer et al. (2023) in building the encoding of a new constituent by using max over the embeddings of the children. This is simpler and more effective than the Bi-LSTM labeled with the subtree used in Dyer et al. (2016) and Liu and Zhang (2017).

Pretrained Embeddings. In (Liu and Zhang, 2017), the standard at the time was to use pretrained word embeddings such as those from word2vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014), or fasttext (Bojanowski et al., 2017). Later work incorporated character models (Akbik et al., 2018). The well-known Berkeley Neural Parser uses XLNet (Yang et al., 2019). We find that for English, Electra-Large (Clark et al., 2020) works best of the currently available pretrained word embeddings. Following best practices, we finetuned the embeddings and transformers for the parsing task. For release as FOSS, we found that PEFT, from Mangrulkar et al. (2022), reduces download size for a single model and VRAM when used in an ensemble without sacrificing model quality.

Attention Layer. Multiple past works on CKY constituency parsing add a specialized attention layer after the transformer, such as Partitioned Attention (Kitaev and Klein, 2018), Label Attention (Mrini et al., 2020), and Bi-Directional Attention (Kim et al., 2023).

Here, we combine ideas from Bi-Directional Attention with Streaming LLM (Xiao et al., 2024). Bi-Directional attention computes attention layers twice, alternatively masking forwards and backwards directions. Streaming LLM uses a windowed attention layer with attention sinks to improve NLG. We find that a single layer of windowed attention in both the forwards and backwards directions improves scores on lower resource constituency datasets, with insignificant effects on larger datasets (table 2).

Lang	No Attn	Attn
EN	96.40	96.37
ZH	95.16	95.07
DE	95.68	95.77
ID	89.22	89.28
IT	83.68	83.87
JA	93.06	93.28
VI	82.80	83.30

Table 2: Dev set scores w/ and w/o attn layer

Lang	No Attn	Attn
EN	95.46	95.51
ZH	94.19	94.49
DE	94.09	94.27
ID	88.76	89.03
IT	83.41	83.59
JA	92.06	92.31
VI	82.33	82.86

Table 3: Dev scores w/ and w/o attn, 5000 sentences

To explore how the resource size affects these scores, we reran the experiment with a randomly selected 5000 sentences for each language (table 3). In this case, the attention layer has a much more pronounced effect. We hypothesize that more data enables the LSTM layer of the encoder to capture the same information extracted by the attention layer in lower data settings.

Ensemble. We investigated the gains that are possible by ensembling multiple parsing models (at the cost of more computation). For chart parsing, a common technique is to use a high-accuracy language model to rerank many outputs of a single parser model, as in (Choe and Charniak, 2016). For transition-based parsing, there is a simpler mechanism for combining the results of multiple models. Instead of choosing a transition by taking the maximum scoring legal transition from one model, we first add the logits from several models, then use the transition with the maximum sum.

It is necessary to build models with different results in order to successfully ensemble them. We explore three methods of achieving the necessary differentiation: (i) initializing each model's layers with a different random seed (*varied seed*), (ii) using different numbers of layers from the transformer for the input embedding (*mixed layers*), or (iii) randomly reweighting the training trees while keeping the structure the same, ensuring training proceeds differently (*random reweighting*). Each method produced useful gains versus a single model, without much or consistent differentiation between them; see table 4.

Model	ID Icon	DE Tiger
single model	89.12	95.76
mixed layers ensemble	89.86	95.93
varied seed ensemble	89.74	95.95
random reweighting ensemble	89.82	95.89

Table 4: When combining models to make an ensemble, different methods for building base models in the ensemble have roughly equivalent dev scores

4 Training Improvements

Oracle. The standard oracle training for transition parsers is teacher forcing: backpropagate the errors from the prediction of a transition, apply the correct transition, and repeat. The limitation of this approach is, at runtime, the model will naturally make errors, resulting in a state which does not directly correspond to any training state.

An improvement is to update the remaining transitions to build the best result still possible after an error. This technique is a *dynamic oracle*, introduced in Goldberg and Nivre (2012) for dependency parsers. This technique works for constituency parsing as well, as shown in the bottom-up parser of Fernández-González and Gómez-Rodríguez (2019) or the discontinuous parser of Coavoux and Cohen (2019).

Fernández-González and Gómez-Rodríguez (2018) further explored this concept for top-down and in-order parsers. After each training error, the oracle used the first possible transition which minimized subsequent errors in the tree. However, this is a potentially ambiguous solution, as some transition sequences have multiple fixes which result in the same number of errors.

An example of an ambiguous transition is the subtree in figure 1 from Marcus et al. (1993). In both the in-order and top-down schemes, the correct transition is to close the ADJP after the word "rolled". If the Close is missed, the dynamic oracle can correct the sequence equally well by closing after "sheet" or after "steel".

We ran an experiment using multiple different types of ambiguous top-down dynamic oracle repairs (see Appendix E). When run over multiple languages, we found that not using a dynamic oracle when the optimal repair is ambiguous works better than choosing either randomly or deterministically; see table 5. Attempting to continue using the dynamic oracle by using the first valid repair, the last valid repair, or using the model itself to predict the repair were each less effective than using teacher forcing specifically for the ambiguous errors.

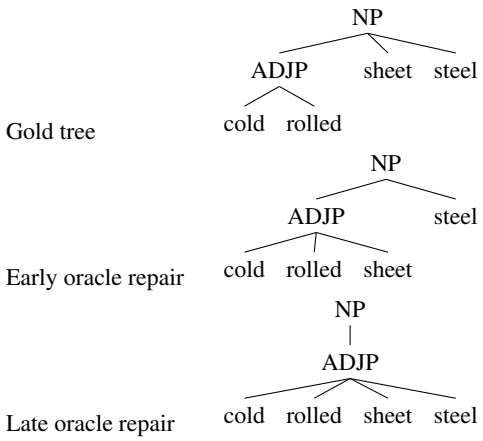


Figure 1: Example of an ambiguous repair

Lang	no oracle	w/o amb	w/ amb
EN (no trans)	92.24	92.68	92.68
ZH	90.96	91.41	91.37
DE	95.76	95.81	95.79
ID	88.75	89.13	88.89
IT	83.62	84.00	84.04
VI	82.60	82.92	82.90
avg Δ	−0.34		−0.05

Table 5: Dev set scores for the top-down model show that leaving out ambiguous repairs is slightly helpful

Self-Training. We generalized the methods of McClosky et al. (2006) and Choe and Charniak (2016) to build a corpus of silver trees for self-training for each of the languages studied. Those works used multiple types of parsers to find higher quality parses of a large corpus of newswire. In this work, we use active learning to select higher quality silver trees.

The inspiration for our method comes from Swayamdipta et al. (2020) and Karamcheti et al. (2021), who explore the idea that a high quality dataset has two general traits, high accuracy and high difficulty.

To produce silver parses for a given language, we use two ensembles (see 3), one of top-down parsers and the other of in-order parsers, and parse Wikipedia for that language. By only keeping the trees where the two ensembles agree, although not guaranteed to filter only correct parses, we build a dataset of silver trees with higher accuracy than either ensemble can produce by itself.

Such a silver dataset does not use the idea of difficulty from the active learning work, though. We can extend this by noting that the individual parsers

in those ensembles do not always agree, with less agreement occurring on "harder" parses. For each silver parse, we tally how many of those individual parsers return that exact parse. Using only the parses with the *least* agreement in an ensemble produces a silver dataset which is both accurate and difficult, improving the overall results. As shown in figure 2, using trees with full agreement produced little effect, and the dataset with zero or one agreeing parsers were too small to use by themselves.[1][2] Table 6 shows the fraction of trees built from Wikipedia which had all 10 models agree, along with the improvement after discarding the full agreement trees.

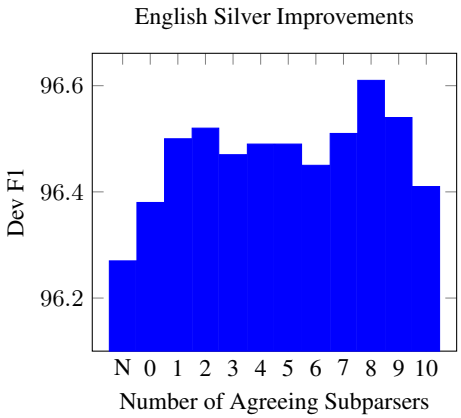


Figure 2: All silver datasets showed an improvement compared to the baseline N, but the smallest one was too small and the full agreement portion was too easy

Lang	Agreement	W/O	With
EN	0.688	96.40	96.61
ZH	0.208	93.25	93.50
DE	0.626	93.06	93.34
ID	0.419	88.68	89.01
IT	0.305	84.18	84.80
JA	0.257	93.28	93.43
VI	0.325	78.91	79.66

Table 6: In-order test scores improve for each dataset with added silver data

5 Results

In table 7, we test these techniques on 7 languages to explore a mixture of word orders and dataset sizes. For English, we investigate multiple transformers to make a fair comparison to past work, and find that Electra gives the best result for this model. We find that ensembling gives an average gain of 0.56 F1, and self training gives an average gain of 0.37 F1. Using an approximate randomization test[3], each of leaving out ambiguous oracle repairs and the windowed attention layer give statistically significant gains with $p < 0.05$.

en_ptb3		
Kim et al. (2023)	Bert	95.86
Yang and Tu (2022)	Bert	96.01
Ours	Bert	95.95
Ours - ensemble	Bert	96.13
Kim et al. (2023)	XLNet	96.47
Ours	Electra	96.40
Ours - ensemble	Electra	96.61
Ours - self	Electra	96.61
Ours - self ensemble	Electra	96.70
zh_ctb51		
Kim et al. (2023)	Bert	94.15
Ours	Electra	93.25
Ours - ensemble		93.71
Ours - self		93.50
Ours - self ensemble		93.66
de_tiger		
Kitaev et al. (2019)	XLM-R	92.10
Ours	Electra	93.06
Ours - ensemble		93.47
Ours - self		93.34
Ours - self ensemble		93.66
id_icon		
Suan Lim et al. (2023)	IndoSpanBERT	88.85
Ours	Indobert	88.68
Ours - ensemble		88.99
Ours - self		89.01
Ours - self ensemble		89.45
it_vit		
Ours	Electra	84.18
Ours - ensemble		84.97
Ours - self		84.80
Ours - self ensemble		85.35
ja_alt		
Ours	Roberta	93.28
Ours - ensemble		94.06
Ours - self		93.43
Ours - self ensemble		94.12
vi_vlsp22		
Bauer et al. (2023)	PhoBert	78.73
Ours	PhoBert	78.91
Ours - ensemble		79.90
Ours - self		79.66
Ours - self ensemble		80.18

Table 7: Bracket F1 for the in-order model compared to previous results. Individual scores are the average of 5 models, and ensembles are those 5 models ensembled together. EN with Bert is included to provide a fair comparison to previous work with bert-large. ZH does not show an improvement compared to previous work. DE, ID, and VI all demonstrate large improvements. For notes on the datasets and the transformers used, see Appendix C

[1] It is possible for zero parsers to agree if each sub-model disagrees with the other nine on different transitions.

[2] This experiment in other languages showed the same effect.

[3] https://github.com/Sleemanmunk/approximate-randomization

This work sets a new SOTA score for a single model parser for PTB when using self-training, while performing close to SOTA for CTB. It also outperforms previously established best results for German, Indonesian, and Vietnamese.

6 Conclusion

In this paper, we have shown the continuing viability of neural transition-based constituency parsing, once the basic technique is carefully combined with state-of-the-art neural text encoders, other successful parser techniques, and active learning.

Limitations

Our parser only supports continuous (projective, tree-structured) constituency parses. There are transition schemes which address discontinuous trees. For example, Coavoux and Cohen (2019) wrote a parser in 2019 which achieved (at the time) SOTA results on discontinuous English and German treebanks. This limitation affects the German results in particular, where we use the SPMRL continuous version of discontinuous constituencies rather than the original Tiger trees.

The running time advantage of a transition parser is that it theoretically runs in $O(N)$ time, as opposed to a chart parser which needs $O(N^3)$ time. Once a transformer with full sentence context is used as the input embedding, the time complexity of the model becomes $O(N^2)$ at best. However, in practice, this model is actually slower than a modern chart parser. There is a large constant factor cost in the usage of Python control code to determine which operations to perform next. Whether this is because of a limitation inherent to Pytorch, an unavoidable limitation of determining at runtime which operations to perform rather than batching every operation at once, or a deficiency in the implementation is an open question.

We compare empirical results to several existing models, showing competitive or SOTA results for multiple languages. However, there are several continuous datasets not tested here, including but not limited to other treebanks in SPMRL, multiple Spanish resources from LDC, and the Icelandic Parsed Historic Corpus (Rögnvaldsson et al., 2012). Expanding the language suite used for testing would provide further evidence of the usefulness of the techniques described here and would improve the usability of this parser. Models for additional tasks will be provided upon request.

Ethics Statement

The model described above has all of the ethical limitations of the underlying datasets. It will parse any amount of offensive, misleading, or otherwise inappropriate text without any hesitation.

Constituency parsing is a highly specialized field, and as such, lower resourced languages are less studied. We deliberately chose two examples of lower resourced languages (Indonesian and Vietnamese) to highlight recent work by those communities.

While training a single parser is not expensive, the experiments described in this paper involved training many parsers to verify average results instead of using a single result, costing many days of GPU time. The data center used for the training is powered by renewable energy, mitigating the environmental impact of this work.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable feedback. Hung Bui and Vy Thai contributed valuable experiments to past versions of this work (Bauer et al., 2023). We also thank Sidd Karamcheti and Drew Hudson for valuable conversations about the parser in its early stages. Finally, we thank the licensees of CoreNLP for their continued support of StanfordNLP, which provided funding for this work.

References

Julien Abadji, Pedro Ortiz Suarez, Laurent Romary, and Benoît Sagot. 2022. Towards a Cleaner Document-Oriented Multilingual Crawled Corpus. *arXiv e-prints*, arXiv:2201.06642.

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. FLAIR: An easy-to-use framework for state-of-the-art NLP. In *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Linda Alfieri and Fabio Tamburini. 2016. (almost) automatic conversion of the venice italian treebank into the merged italian dependency treebank format. In *Proceedings of the Third Italian Conference on Computational Linguistics CLiC-it 2016*, pages 19–23.

John Bauer, Hung Bui, Vy Thai, and Christopher Manning. 2023. In-order transition-based parsing for vietnamese. *Journal of Computer Science and Cybernetics*, 39(3):207–221.

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.

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2001. The tiger treebank. In *Third Workshop on Linguistically Interpreted Corpora LINC-2001*, Leuven, Belgium.

Do Kook Choe and Eugene Charniak. 2016. Parsing as language modeling. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2331–2336, Austin, Texas. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2016. Fast and accurate deep network learning by exponential linear units (ELUs). In *ICLR*. arXiv.

Maximin Coavoux and Shay B. Cohen. 2019. Discontinuous constituency parsing with a stack-free transition system and a dynamic oracle. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 204–217, Minneapolis, Minnesota. Association for Computational Linguistics.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Shijin Wang, and Guoping Hu. 2020. Revisiting pre-trained models for Chinese natural language processing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 657–668, Online. Association for Computational Linguistics.

Rodolfo Delmonte, Antonella Bristot, and Sara Tonelli. 2007. VIT – Venice Italian Treebank: Syntactic and quantitative features. In *Proceedings of the Sixth International Workshop on Treebanks and Linguistic Theories.*

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. 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 334–343, Beijing, China. Association for Computational Linguistics.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209, San Diego, California. Association for Computational Linguistics.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2018. Dynamic oracles for top-down and in-order shift-reduce constituent parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1303–1313, Brussels, Belgium. Association for Computational Linguistics.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2019. Faster shift-reduce constituent parsing with a non-binary, bottom-up strategy. *Artificial Intelligence*, 275:559–574.

Filip Ginter, Jan Hajič, Juhani Luotolahti, Milan Straka, and Daniel Zeman. 2017. CoNLL 2017 shared task -

automatically annotated raw texts and word embeddings. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Yoav Goldberg and Joakim Nivre. 2012. A dynamic oracle for arc-eager dependency parsing. In *COLING*.

My Linh Ha, Thi Minh Huyen Nguyen, The Quyen Ngo, Tuan Thanh Le, Tran Thai Dang, Viet Hoang Ngo, Xuan Dung Doan, Thi Luong Nguyen, Van Cuong Le, Thi Hue Phan, and Xuan Luong Vu. 2022. VLSP 2022 Challenge: Vietnamese Constituency Parsing. In *Journal of Computer Science and Cybernetics*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

Siddharth Karamcheti, Ranjay Krishna, Li Fei-Fei, and Christopher Manning. 2021. Mind your outliers! Investigating the negative impact of outliers on active learning for visual question answering. 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)*, pages 7265–7281, Online. Association for Computational Linguistics.

Soohyeong Kim, Whanhee Cho, Minji Kim, and Yong Choi. 2023. Bidirectional masked self-attention and n-gram span attention for constituency parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 326–338, Singapore. Association for Computational Linguistics.

Nikita Kitaev, Steven Cao, and Dan Klein. 2019. Multilingual constituency parsing with self-attention and pre-training. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3499–3505, Florence, Italy. Association for Computational Linguistics.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686, Melbourne, Australia. Association for Computational Linguistics.

Fajri Koto, Afshin Rahimi, Jey Han Lau, and Timothy Baldwin. 2020. IndoLEM and IndoBERT: A benchmark dataset and pre-trained language model for Indonesian NLP. In *Proceedings of the 28th COLING*.

Jiangming Liu and Yue Zhang. 2017. In-order transition-based constituent parsing. *Transactions of the Association for Computational Linguistics*, 5:413–424.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. *Preprint*, arXiv:1711.05101.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. 2022. Peft: State-of-the-art parameter-efficient fine-tuning methods. `https://github.com/huggingface/peft`.

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

Mitchell P. Marcus, Beatrice Santorini, Mary Ann Marcinkiewicz, and Ann Taylor. 1999. Treebank-3 ldc99t42. Philadelphia: Linguistic Data Consortium.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Reranking and self-training for parser adaptation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 337–344, Sydney, Australia. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *Proceedings of Workshop at ICLR*.

Khalil Mrini, Franck Dernoncourt, Quan Hung Tran, Trung Bui, Walter Chang, and Ndapa Nakashole. 2020. Rethinking self-attention: Towards interpretability in neural parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 731–742, Online. Association for Computational Linguistics.

Dat Quoc Nguyen and Anh Tuan Nguyen. 2020. PhoBERT: Pre-trained language models for Vietnamese. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1037–1042.

Minh Van Nguyen, Viet Dac Lai, Amir Pouran Ben Veyseh, and Thien Huu Nguyen. 2021. Trankit: A light-weight transformer-based toolkit for multilingual natural language processing. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the Eighth International Conference on Parsing Technologies*, pages 149–160, Nancy, France.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, and 2 others. 2019. Pytorch: An imperative style, high-performance deep learning library. *Preprint*, arXiv:1912.01703.

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.

Eiríkur Rögnvaldsson, Anton Karl Ingason, Einar Freyr Sigurðsson, and Joel Wallenberg. 2012. The Icelandic parsed historical corpus (IcePaHC). In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 1977–1984, Istanbul, Turkey. European Language Resources Association (ELRA).

Kei Sawada, Tianyu Zhao, Makoto Shing, Kentaro Mitsui, Akio Kaga, Yukiya Hono, Toshiaki Wakatsuki, and Koh Mitsuda. 2024. Release of pre-trained models for the Japanese language. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 13898–13905. https://arxiv.org/abs/2404.01657.

Djamé Seddah, Reut Tsarfaty, Sandra Kübler, Marie Candito, Jinho D. Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola Galletebeitia, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiórkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, and 4 others. 2013. Overview of the SPMRL 2013 shared task: A cross-framework evaluation of parsing morphologically rich languages. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 146–182, Seattle, Washington, USA. Association for Computational Linguistics.

Ee Suan Lim, Wei Qi Leong, Ngan Thanh Nguyen, Dea Adhista, Wei Ming Kng, William Chandra Tjh, and Ayu Purwarianti. 2023. ICON: Building a large-scale benchmark constituency treebank for the Indonesian language. In *Proceedings of the 21st International Workshop on Treebanks and Linguistic Theories (TLT, GURT/SyntaxFest 2023)*, pages 37–53, Washington, D.C. Association for Computational Linguistics.

Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A. Smith, and Yejin Choi. 2020. Dataset cartography: Mapping and diagnosing datasets with training dynamics. *Preprint*, arXiv:2009.10795.

Heike Telljohann, Erhard W. Hinrichs, Sandra Kübler, Heike Zinsmeister, and Kathrin Beck. 2012. Stylebook for the Tübingen treebank of written german (TüBa-D/Z). Technical report, University of Tübingen, Seminar für Sprachwissenschaft.

Ye Kyaw Thu, Win Pa Pa, Masao Utiyama, Andrew Finch, and Eiichiro Sumita. 2016. Introducing the Asian language treebank (ALT). In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1574–1578, Portorož, Slovenia. European Language Resources Association (ELRA).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Preprint*, arXiv:1706.03762.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient streaming language models with attention sinks. *Preprint*, arXiv:2309.17453.

Naiwen Xue, Fei Xia, Fu-Dong Chiou, and Marta Palmer. 2005. The penn chinese treebank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 11(2):207–238.

Kaiyu Yang and Jia Deng. 2020. Strongly incremental constituency parsing with graph neural networks. In *Neural Information Processing Systems (NeurIPS)*.

Songlin Yang and Kewei Tu. 2022. Bottom-up constituency parsing and nested named entity recognition with pointer networks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2403–2416, Dublin, Ireland. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: generalized autoregressive pretraining for language understanding. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, Red Hook, NY, USA. Curran Associates Inc.

Daniel H. Younger. 1967. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10(2):189–208.

Matthew D. Zeiler. 2012. Adadelta: An adaptive learning rate method. *Preprint*, arXiv:1212.5701.

Junru Zhou and Hai Zhao. 2019. Head-Driven Phrase Structure Grammar parsing on Penn Treebank. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2396–2408, Florence, Italy. Association for Computational Linguistics.

A Transition Parsing and Related Work

The underlying mechanism of transition-based parsing is similar to that of a shift/reduce compiler.

To implement a shift/reduce compiler without relying on the call stack, the compiler maintains a state with two data structures: a stack of components it has already built, which will have zero or more pieces on it, and a queue of tokens remaining to be processed. The operations allowed are to shift a new item from the queue onto the stack and to reduce some number of items from the stack into a larger, combined item.

This same general idea applies to transition parsing for NLP models. However, rather than deterministic rules to choose the next action, the parser models the current state to choose the next action. The training mechanism is to turn the gold parse tree into a gold sequence of actions, then train the model to correctly predict those actions.

The first transition models to use this mechanism focused on dependencies, becoming a well-studied method for dependency parsing (Nivre, 2003).

In one early work using this type of model for constituencies, the transitions chosen either combine the most recent pair of subtrees into a larger subtree, or create a subtree of one word out of the next node. Described as a 'bottom up' model, this constructs a binarized parse tree out of the entire sentence. Adding labels to the reduce actions makes a labeled tree, and further adding 'complete' or 'incomplete' state to the reduce actions allows the model to construct a tree with arbitrary branching, not just binarized (Clevert et al., 2016).

Building trees in a top-down manner allows for a more comprehensive understanding of the sentence or local phrase when making a determination of the next action. Adding small recurrent networks, in particular an LSTM over the subtrees, facilitated recusively constructing embeddings for phrases as well as words. Adding these mechanisms was, at the time, very close to state of the art (Dyer et al., 2016).

The in-order transition sequence improves on this mechanism by delaying when the labeling of the subtree is chosen. First, the left side of the subtree is built, and only after that is built does the model label the subtree it is currently building (Liu and Zhang, 2017). Predicting the label of the constituent with the additional information of the first child further improves accuracy.

Aside from these models, there are even more complicated options available. For example, some formalisms allow for discontinuous parse trees to represent long distance dependencies, and specialized action sets exist to account for those (Coavoux and Cohen, 2019). Furthermore, recent work explored using attention to attach the currently built subtree at any depth of the previously built subtrees, not just the top, a mechanism titled attach-juxtapose (Yang and Deng, 2020).

In this work, we continue with the top-down and in-order sequences.

B Hyperparameters and Training Setup

Encoding	
Embedding Dim	100
Forward & Backward CharLM	1024 + 1024
Electra	768
POS Tag Embedding	20
LSTM Layers	2
LSTM Input Dropout	0.2
Prediction Head	
MLP Layers	3
Nonlinearity	ReLU
Optimization	
Batch Size	50 trees
Eval Frequency	5000 trees
AdaDelta optimization	200 evals
AdaDelta LR	1.0
AdaDelta WD	0.02
AdamW optimization	200 evals
AdamW LR	0.0002
AdamW WD	0.05
Plateau LR Decay	0.6
Plateau Patience	5
Plateau Cooldown	10

Table 8: Hyperparameters used when training the model. The publicly released software includes flags for each of these settings.

After 200 training and evaluation cycles with AdaDelta (Zeiler, 2012), the model with the best dev set evaluation is trained for another 200 cycles with AdamW (Loshchilov and Hutter, 2019). The model with the best dev set evaluation among all of the models trained is kept as the final model.

The final model size is dependent on whether the model uses a transformer and whether that transformer is fully finetuned or finetuned with LoRA (Hu et al., 2021). A fully finetuned English model

using ELECTRA has 59,645,169 parameters, of which 25,165,824 were finetuned Electra parameters.

This work used torch version 2.3.0 and peft version 0.10.0 (Paszke et al., 2019; Mangrulkar et al., 2022). It is likely earlier or later versions of both packages would produce similar results.

Fully training a parser with these hyperparameters takes 1–2 days on a consumer GPU, such as an Nvidia RTX 3090, depending on the dataset used. German, for example, trains faster than Chinese, as the German treebank has a much shorter average sentence length.

When building a silver dataset, we randomly chose 200,000 trees from the set of trees which did not have full agreement between the 10 submodels.

Aside from the silver dataset improvement graphs, results reported in this paper use an average of 5 models with different random seeds to report single model results. Ensemble results are based on a single ensemble built from those 5 models.

C Datasets

C.1 English

For English, we evaluate on the standard Penn Treebank (Marcus et al., 1993) We use the standard Evalb evaluation,[4] with PRT/ADVP collapsing and punctutation removal, to measure the performance of the parser. In particular, we use the "nk.prm" settings from (Kitaev and Klein, 2018) with the standard EvalB metric. For the input embedding, we use Electra-Large (Clark et al., 2020), along with a charlm (Akbik et al., 2019) and the word vectors from the CoNLL 2017 shared task (Ginter et al., 2017).

For some of the experiments, we built the model without the pretrained transformer or charlm in order to better emphasize the difference in model quality the proposed changes made. For example, the silver experiment in section 4 has results with and without the more powerful representations.

Previous work has used Devlin et al. (2018) and Yang et al. (2019) as the embedding for SOTA results. We compare with Bert for a fair comparison, while finding that XLNet and other autoregressive models are less compatible with transition constituency parsing, perhaps because the bidirectional encoders are necessary to have proper knowledge of future words.

[4] https://nlp.cs.nyu.edu/evalb/

C.2 Chinese

The Chinese Treebank version 5.1 is another standard measurement for the quality of a model (Xue et al., 2005).

One caveat with CTB is there are two standard test splits of CTB 5.1 in the literature. One is a split which includes new trees from CTB 5.1, used as recently as (Zhou and Zhao, 2019), and the other is to inherit the smaller test section from previous versions of CTB, such as used in (Kim et al., 2023). Experimentally, the smaller test set is entirely newswire and models achieve higher scores on the more structured language. Unfortunately, this dichotomy has previously gone unreported, and CTB constituency leaderboards tend to mix scores from the two test sets.[5]

In this work, we use the split of 301–325 for dev, 271–300 for test, and ignore 400–439 to provide a fair comparison with the most recent SOTA, Kim et al. (2023). The chinese-electra-180g-large-discriminator transformer (Cui et al., 2020) produced the best results for us, as opposed to Kim et al. (2023), which used Chinese BERT.

C.3 German

Three commonly used treebanks exist for German: Negra, Tiger, and Tübingen (Brants et al., 2001; Telljohann et al., 2012). Licensing reasons prevented us from using Tübingen, as we intend to publicly release our models, and Negra was included in the Tiger treebank. Accordingly, we used Tiger as the best option available to us.

For this paper, we use the SPMRL version of the dataset and evaluate it with the "spmrl.prm" settings for EvalB (Seddah et al., 2013). In order to compare the accuracy of the parser without concern for the tokenization, we use the gold tokenization provided in the SPMRL task. Furthermore, while the original Tiger treebank uses discontinuous trees, this parser only handles continuous trees. The SPMRL version of the treebank allows for such processing.

We use the Electra model from german-nlp-group[6] to build the final version of the parser. The previous SOTA on this dataset, the Kitaev et al. (2019) parser, used XLM-R (Conneau et al., 2020).

SPMRL includes several other treebanks. Models for those tasks will be made available for com-

[5] https://chinesenlp.xyz/docs/constituency_parsing.html

[6] https://huggingface.co/german-nlp-group/electra-base-german-uncased

parison or annotation purposes on request.

C.4 Indonesian

At GURT 2023, Suan Lim et al introduced a newly written Indonesian constituency treebank (Suan Lim et al., 2023). They reported a score of 88.85 using Benepar and a custom fine tuned transformer, using the standard nk.prm parameters from evalb. We built a character model from Wikipedia and the Oscar Common Crawl (Abadji et al., 2022), then tested on the various publicly available Indonesian transformers on HuggingFace.

Our best models use the Indolem Indobert model (Koto et al., 2020); other Indonesian or multilingual transformers were less accurate in our experiments.

C.5 Italian

We use the Venice Treebank (Delmonte et al., 2007) to build an Italian model. As the original treebank does not have defined train/dev/test splits, we align the sentences with the edited sentences of the UD conversion of VIT (Alfieri and Tamburini, 2016), which does have train/dev/test splits. Where no alignment is possible, such as for sentences which are split in the UD dataset, we drop the sentence. This leaves 7875 train, 683 dev, and 1042 test trees. Code to reproduce this split is included in the software release.

Using the Electra model from DBMDZ[7] produced the most accurate model for this task.

We evaluated this model using the standard evalb evaluation, adding *punto* to the list of ignored constituencies as that represents punctuation in this treebank.

C.6 Japanese

We use the Japanese ALT (Thu et al., 2016) to build a Japanese model. This is a parallel treebank, intended to eventually have many languages parsed, but currently only Japanese is finished enough to use for constituency parsing. The treebank advertises 20,106 trees, but some number are missing from the Japanese portion of the corpus. We further eliminate 9 trees for having entire words composed of nothing but spaces. This leaves 17195 train, 934 dev, and 931 test trees.

After some brief exploration, we found that the Rinna Roberta model (Sawada et al., 2024) was a good combination of ease of use and performance.

[7] https://huggingface.co/dbmdz/electra-base-italian-xxl-cased-discriminator

C.7 Vietnamese

In 2022, VLSP produced a constituency parsing dataset, along with a bakeoff (Ha et al., 2022). We compare our results against the best performing model from the bakeoff, from (Bauer et al., 2023). Note that the publicly reported contest scores include both POS and bracket scores, whereas this score is reported on only brackets, leading to our scores differing from the publicly reported scores. We use evalb to evaluate, adding *punct* to nk.prm.

As there is no defined dev set for VLSP22, we use a random sample of 1/10th of the training dataset.

We found the best transformer for building the parser was Phobert-Large (Nguyen and Nguyen, 2020).

C.8 Availability

The previous artifacts are available for research purposes. PTB and CTB are both available from LDC, whereas VIT is available at ELRA. SPMRL and Vietnamese were provided by request. ICON and ALT are freely available as part of the published work. Whenever possible, we confirmed with the authors that models derived from the work can be publicly released. This limitation informed our choice of German treebank, in particular.

The models derived from these datasets will be available at (anonymous) under the Apache License, Version 2.0.

D Example Oracle Sequence

Both the in-order and top-down transition scheme are capable of constructing any tree, doing so with an unambiguous transition sequence. Included in Table 9 is an end to end parsing example for "Transition parsing is fun" using a top-down transition sequence.

E Ambiguous Dynamic Oracle

An unambiguous dynamic oracle repair is when the dynamic oracle has only one minimum error option for how to rewrite the transition sequence after the parser makes an error at training time. An example of this is when the top-down gold sequence is to Shift, but the model chooses to Close. The incorrectly closed bracket is both a recall and a precision error, but causes no further errors provided the remainder of the sequence is properly followed, so the best, unambiguous repair is to remove the correct Close from later in the sequence.

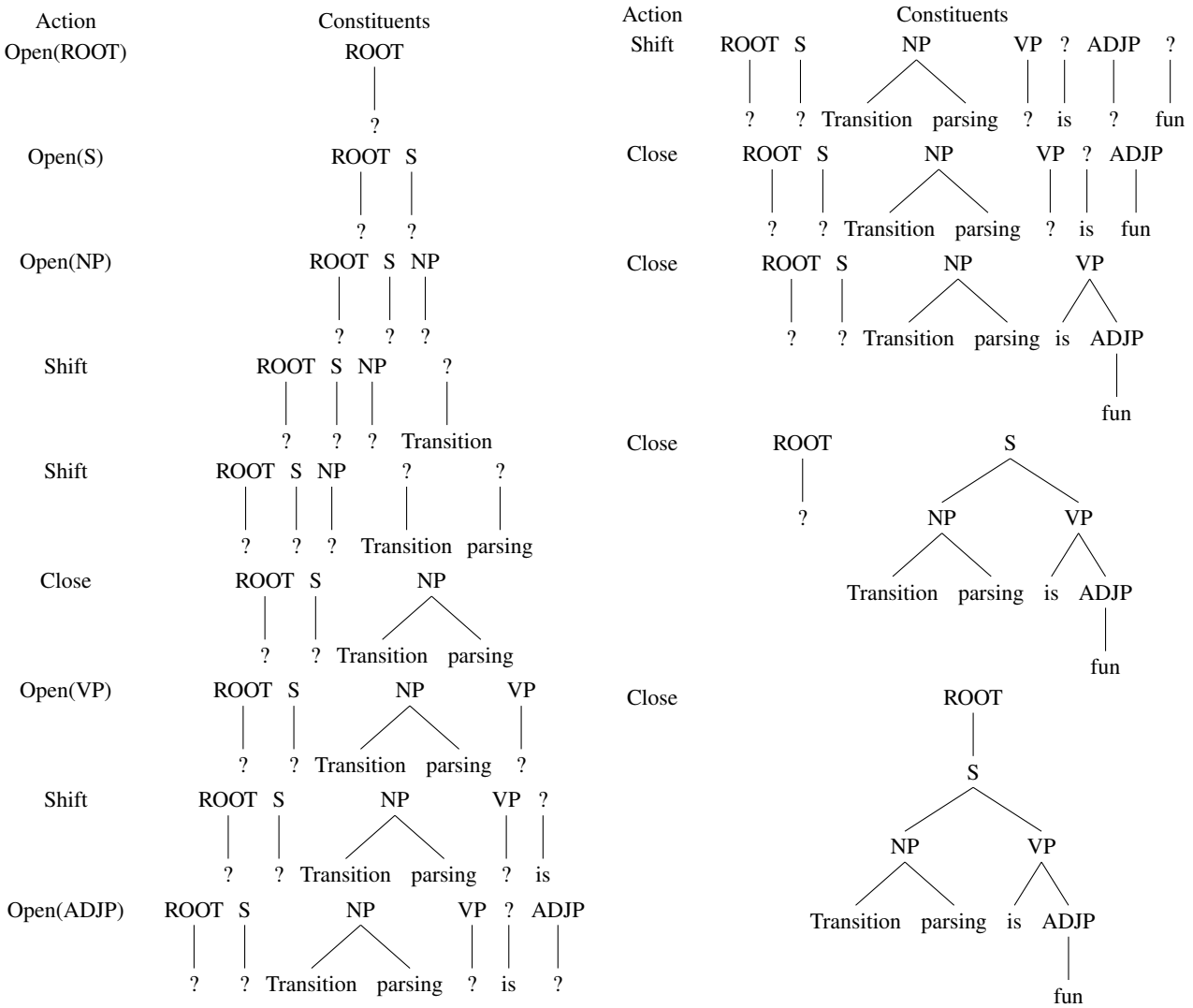
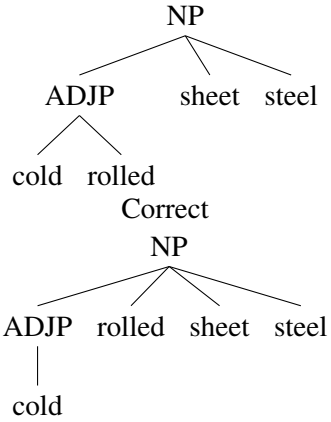


Table 9: Complete top-down transition sequence for “Transition parsing is fun”



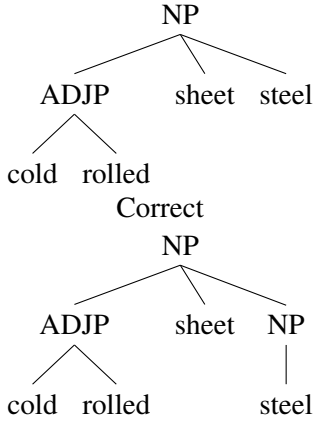
After an incorrect Close, this is the best tree the model can construct

An ambiguous error has multiple possible resolutions which produce the same number of errors.

The most common ambiguous transition error for the in-order model is that of a shift replaced with an open, empirically representing 1/6th of the errors made by a fully trained model. Such a transition introduces a single precision error, the incorrectly opened bracket. There are multiple ways of "repairing" the transition sequence after such an error. For example, an immediate close of the new bracket represents a unary transition around the previous item, whereas a close at the end of the bracket could contain several subtrees.

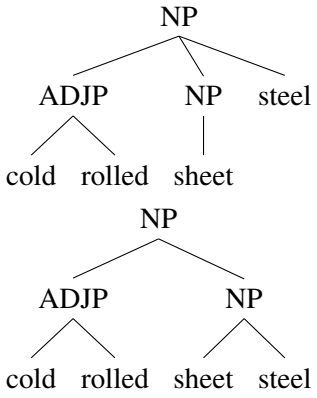
In this section, we enumerate the ambiguous errors possible in the top-down scheme. The in-order scheme is similar, but with more complicated exceptions.

If the correct transition is Shift, but the model predicts Open, this creates a new bracket where there is none, a single precision error. The new constituent must be closed at some point. If the Shift was the final word of the current bracket, this is not ambiguous:

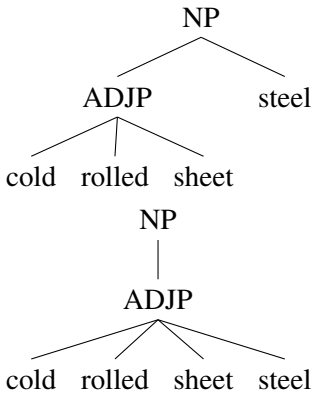


Incorrect NP with an unambiguous close

If the incorrect NP in this example opens before the word 'sheet', though, the new constituent can close either after 'sheet' or 'steel':

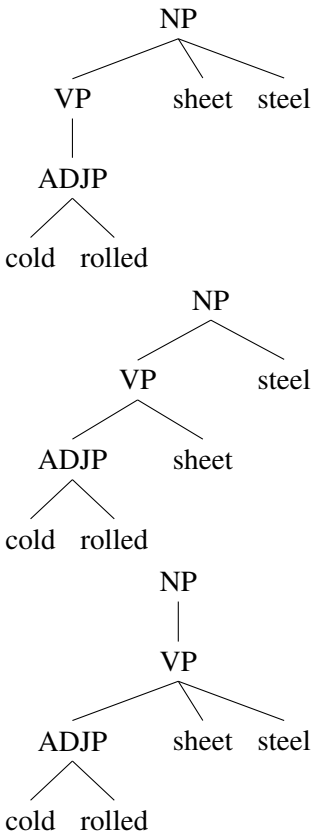


If the correct transition was to Close, but the model predicts Shift, this causes both a precision error and a recall error. The newly created bracket is a precision error, and the correct bracket cannot be recovered, a recall error. The dynamic oracle repair is ambiguous in the case of more than one piece after which the bracket could close. Continuing with the 'steel' example, if the ADJP is not correctly closed:



The same effect happens with a Close to Open error. Both the new constituent and the unclosed constituent from earlier need to be closed, and the time at which to do that is ambiguous unless the constituent has only one possible resolution. Note that the incorrect Open does not necessarily cause a recall error in the case of building a correct constituent at the wrong time, such as in the phrase 'eat (NP spaghetti) (PP with a fork)' becoming 'eat (NP spaghetti (PP with a fork))'

In the case of an Open constituent incorrectly labeled, in most cases this is a precision error which the dynamic oracle can repair by building the correct constituent anyway, then closing the incorrect constituent later. When to close may be ambiguous, such as if the model added an incorrect VP in the 'cold rolled sheet steel' example:



Another ambiguity in the mislabeled Open case can arise when the incorrectly predicted Open transition is the next correct transition, such as if the model had predicted ADJP instead of NP. In that case, discarding the Open(NP) and the corresponding Close would cause a single recall error instead of a precision error.

F Dynamic Oracle Repair

We ran an extensive test on the in-order transition error described in section 4, when the gold sequence has a Close-Shift, but the prediction is Shift. In this case, the correct bracket is lost, meaning a recall error, and a new, wider bracket is created, meaning a precision error. To prevent further errors, this new bracket must be closed before the bracket enclosing it is also closed. In the case of a bracket wider than one more node, this close transition could occur after any number of additional nodes without further affecting the score, meaning it is an ambiguous oracle repair.

The signal is not strong, but as shown in table 10, the overall trend for this one repair is to be slightly more accurate to not choose any ambiguous transition. Further tests using all of the dynamic oracle repairs found that not using any ambiguous repairs was an improvement.

As observed in section 4 for the top-down oracle, when testing multiple such ambiguous repairs at once, the trend is that using teacher forcing for the ambiguous errors is a slight improvement.

Lang	Unamb	Early	Late
EN	92.53	92.59	92.52
ZH	91.55	91.45	91.38
ID	89.21	89.24	89.21
DE	95.66	95.64	95.72
IT	83.76	83.63	83.65
VI	82.85	82.76	82.78

Table 10: Dev scores for the close/shift error described above show this ambiguous error cannot be deterministically resolved in a positive manner.