Lightweight Fine-tuning for Dependency Parsing: An Alternative to Large Language Models

Anonymous ACL submission

Abstract

While large language models (LLMs) have set new benchmarks for dependency parsing, achieving these gains typically requires fine-tuning billions of parameters, making such approaches impractical in resource-constrained environments. For structured prediction tasks, we show that explicit structural constraints can effectively compensate for limited model capacity. We propose a lightweight parsing framework that combines compact encoder-decoder models (with millions rather than billions of parameters) and a task-specific finite-state machine (FSM) to guide decoding. By decomposing parsing into a pipelined prediction of heads and relations, our method enforces structural validity and reduces error propagation. Experiments on Universal Dependencies (UD) demonstrate that our approach achieves competitive accuracy with decoder-only models over $100\times$ larger. These results suggest that incorporating domain-specific structural constraints offers an efficient alternative to indiscriminate model scaling for dependency parsing.

1 Introduction

Large language models (LLMs) have recently achieved strong performance across many NLP tasks, including dependency parsing, where fine-tuning has been shown to match or even surpass traditional transition-based and graph-based parsers across multiple languages. However, such gains come at a substantial computational cost, making training and deployment challenging in resource-constrained settings.

Despite their different formulations, modern dependency parsers share common structural foundations. Transition-based approaches score parsing actions, while graph-based methods assign scores to candidate arcs; in the deep learning era, both paradigms have largely converged on encoder-decoder architectures that combine contextualized sentence representations with structured prediction.

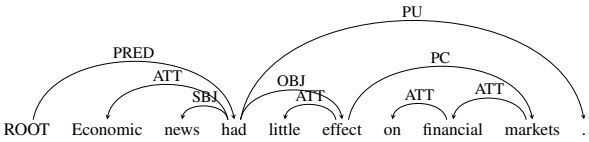

Figure 1: Dependency parse tree for the sentence: *"Economic news had little effect on financial markets."*

In this paper, we propose a lightweight alternative to LLM-based fine-tuning using compact encoder-decoder models with orders of magnitude fewer parameters. Our approach integrates a finite-state machine (FSM) to constrain decoding, ensuring well-formed dependency structures while reducing training and inference costs.

Experiments on multiple Universal Dependencies (UD) treebanks show that, despite using only about one percent of the parameters of recent decoder-only models, our method achieves competitive accuracy across diverse languages. These results suggest that carefully constrained small-scale models can provide an effective and practical solution for structured prediction tasks.

2 Related Work

Early work on dependency parsing primarily relied on statistical approaches. (Nivre, 2003) and (Yamada and Matsumoto, 2003) introduced the first data-driven shift–reduce parsers, and (Nivre, 2004) demonstrated that accurate parses could be obtained in linear time. Within this framework, transition selection was typically formulated as a classification problem. Graph-based approaches soon followed, (McDonald et al., 2005) cast dependency parsing as a maximum spanning tree problem over a fully connected dependency graph, scoring all arcs to obtain the optimal tree, while (Johansson and Nugues, 2008) extended these models with richer feature templates, further improving their accuracy.

The advent of deep learning fundamentally trans-

formed the field. (Chen and Manning, 2014)introduced neural transition-based parsing, using feed-forward networks to boost the accuracy of both transition-based and graph-based approaches beyond statistical baselines. Subsequent research largely focused on neural methods: (Kiperwasser and Goldberg, 2016) demonstrated the strong performance of BiLSTM encoders in both paradigms, while (Dozat and Manning, 2016) proposed the widely adopted biaffine attention mechanism. The introduction of Transformer architectures further improved accuracy, with powerful encoders such as BERT (Devlin et al., 2019) and XLNet (Yang et al., 2019) enabling custom decoders that represent the current state of the art (Mrini et al., 2019; Tian et al., 2022; Zhou and Zhao, 2019).

With the rapid emergence of large language models (LLMs), natural language processing has become increasingly reliant on powerful pretrained encoders and decoders. Fine-tuning general pretrained models on downstream tasks has become the dominant paradigm in many areas. For dependency parsing, (Kondratyuk and Straka, 2019) demonstrated the feasibility of encoder fine-tuning by using multilingual mBERT as the encoder and achieving strong baseline results on Universal Dependencies (UD) treebanks spanning 75 languages. With the fast development of decoder-side LLMs, several approaches have reformulated syntactic parsing as sequence generation rather than scoring transitions or arcs. These methods leverage autoregressive decoders to generate linearized structures or sequences of parsing actions. For instance, (Vasylenko et al., 2023) fine-tuned BART and achieved state-of-the-art results for AMR parsing, while for dependency parsing, (Hromei et al., 2024) and (Matsuda et al., 2025) fine-tuned various decoder-only models on multilingual UD datasets and obtained competitive performance.

Constrained decoding has been widely studied for structured prediction, where constraints are often expressed as finite-state machines and enforced by dynamically restricting the next-token space during decoding (Deutsch et al., 2019; He and Choi, 2023; Geng et al., 2023).

3 Methodology

3.1 Input Representation

Unlike recent approaches that fine-tune large decoder-only language models, we adopt a much lighter encoder–decoder architecture. This choice reduces both training and inference costs while preserving competitive performance.

For input design, we deliberately keep the format minimal and purely structural. On the encoder side, the original sentence is fed directly as a plain sequence, while on the decoder side, we linearize the dependency tree into a simplified CoNLL-U style string containing only `index`, `POS tag`, `head`, and `relation` entries:

```
[ID-1]  [TAG-ADJ]  [ID-2]  [REL-amod]
[ID-2]  [TAG-NOUN]  [ID-3]  [REL-nsubj]
[ID-3] [TAG-VERB] [ID-0] [REL-root] ...
```

We exclude natural-language tokens from the decoder sequence to avoid unnecessary tokenization splits and to emphasize that the task is strictly structured sequence generation, not free-form text generation. This simple representation allows direct and lossless conversion between dependency trees and decoder outputs.

3.2 Pipeline Decoding Strategy

Following (Matsuda et al., 2025), we decompose the decoding into two stages: predicting the head index for each token, and then predicting the dependency relation conditioned on the predicted heads. The decoder therefore receives two consecutive sequences separated by a special token `[SEP]`:

```
[ID-1]   [TAG-ADJ]   [ID-2]   [ID-2]
[TAG-NOUN]  [ID-3]  [ID-3]  [TAG-VERB]
[ID-0]  ...   [SEP]  [ID-1]  [TAG-ADJ]
[ID-2]  [REL-amod]  [ID-2]  [TAG-NOUN]
[ID-3]  [REL-nsubj]  [ID-3]  [TAG-VERB]
[ID-0] [REL-root] ...
```

This staged approach aims to reduce error propagation and improve overall accuracy compared to predicting heads and relations simultaneously. We keep the implementation minimal and consistent with our simplified input format, without introducing additional natural-language prompts or auxiliary tasks.

3.3 Finite-State Machine Constrained Decoding

Prior work on constrained decoding often focuses on generic constraints. In contrast, we design a task-specific FSM tailored to our linearized dependency representation to guarantee well-formed trees.

Autoregressive decoding is widely used for most sequence generation tasks; however, for dependency parsing, it is not ideal because unconstrained

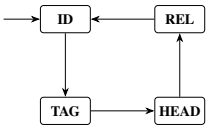

Figure 2: Raw FSM.

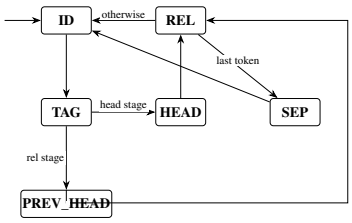

Figure 3: Piped FSM.

generation can easily produce invalid structures, such as repeating POS tags or relations, skipping or duplicating tokens, or generating rootless dependency trees. For example, `[ID-1] [TAG-ADJ] [REL-amod] [REL-nsubj] [ID-2] ...` cannot be mapped to any valid tree because two relations appear without a head prediction.

To address this, we design a finite-state machine (FSM) to regulate decoder outputs. At each decoding step, the FSM enforces permissible token types according to the current state:

- **ID**: force the current word token `[ID-x]`
- **TAG**: force the corresponding POS tag token `[TAG-xxx]`
- **HEAD**: restrict to valid head indices `[ID-0]` (root) through `[ID-n]`
- **REL**: restrict to valid dependency relation tokens `[REL-xxx]`

For the pipeline strategy, we introduce an additional state:

- **PREV_HEAD**: force to reproduce the previously predicted head token for the current word

As shown in Figure 2, our raw decoder cycles through four states (ID, TAG, HEAD, REL) to produce a valid dependency arc for each token. For the pipeline variant (Figure 3), we introduce two additional transitions: PREV_HEAD, which reproduces the previously predicted head, and SEP, which separates head prediction from relation prediction. These additions allow the decoder to reuse intermediate predictions while maintaining structural validity across stages.

This mechanism resembles traditional transition-based parsers, where parsing actions are selected under structural constraints; however, here the FSM serves as a decoding-time guard for a generative model rather than a separate classifier.

By embedding these structural constraints directly into decoding, we ensure that every generated sequence corresponds to a valid dependency tree and reduce error propagation across stages.

3.4 Consensus Decoding

We use beam search in our decoding. While beam search improves sequence quality compared to greedy decoding, it still selects a single sequence that may contain locally optimal but globally suboptimal arcs.

To further stabilize predictions, we adopt a consensus decoding strategy similar to MBR methods used in (Kuncoro et al., 2016; Sagae and Lavie, 2006). This technique applies to both plain autoregressive decoding and our FSM-constrained decoding.

Specifically, from the top-k beam sequences (or dependency trees) $\{T^{(1)}, T^{(2)}, \ldots, T^{(k)}\}$, we construct an arc-weighted graph G over the sentence tokens. For each beam b and each arc $(h \to d, l)$ in $T^{(b)}$, we add a weight $w_b = \frac{1}{1+b}$ in G, adding the weights if the arc appears in multiple beams. Intuitively, higher-ranked beams (lower b) contribute more, while lower-ranked beams still provide useful diversity cues.

Finally, we run a **maximum spanning tree** (MST) algorithm over G to extract the highest-scoring valid dependency tree:

$$T^{\star} = \mathrm{MST}(G).$$

This consensus procedure effectively pools evidence from multiple candidate trees, reduces variance from single-sequence decoding, and mitigates spurious local decisions.

4 Experiments

4.1 Dataset

We evaluate on Universal Dependencies (UD) v2.12, using the official train/dev/test splits. For the initial study, we select five typologically diverse treebanks (`en_ewt`, `de_gsd`, `cs_pdt`, `ko_gsd`, `zh_gsdsimp`) to ensure coverage and allow rapid iteration during early-stage development.

4.2 Baselines

We compare against established systems commonly reported on UD: (1) **UDify** (Kondratyuk and Straka, 2019), a multilingual BERT-based baseline parser, (2) **UDPipe 2.0** (Straka, 2018), top-ranked in

Treebank	en_ewt	de_gsd	cs_pdt	ko_gsd	zh_gsdsimp
% len ≤ 60	99.25	99.48	99.64	100.00	98.15

Table 1: Proportion of Sentences Lengths.

Model	en_ewt	de_gsd	cs_pdt	ko_gsd	zh_gsdsimp
UDify	90.1/88.5	87.8/83.6	94.7/92.9	82.7/74.3	87.9/83.8
UDPipe 2.0	93.4/91.5	89.2/85.5	95.7/94.3	88.9/85.2	86.7/83.6
Matsuda et al.	95.5/94.1	90.3/86.7	94.9/93.3	90.6/87.6	89.7/87.3
Ours (raw)	91.0/85.1	88.5/81.6	93.6/90.7	85.1/83.6	88.8/85.8
Ours (piped)	92.1/89.8	88.2/84.0	94.1/90.5	86.2/84.8	89.2/84.9
Ours (piped+con)	94.1/90.5	88.6/84.3	94.2/91.8	90.4/88.2	89.2/85.3

Table 2: Main Results on UD Test Sets (UAS/LAS)

CoNLL 2018 and (3) (Matsuda et al., 2025),a recent LLM-based sequence-generation approach designed for multilingual dependency parsing. For UDify, we report the numbers published in their original paper. For UDPipe 2.0 and Matsuda et al., we directly cite the scores listed in (Matsuda et al., 2025).

4.3 Setup

We conduct all experiments under a single, fixed setup to ensure comparability.[1]

Due to GPU memory constraints [2], we apply a length-based filtering strategy and remove sentences longer than 60 tokens from all datasets. This filtering is applied uniformly across training, development, and test sets for all models reported in this paper.

Table 1 reports the proportion of sentences with length ≤ 60 for each treebank. While the majority of sentences fall below this threshold, the proportion of longer sentences varies across languages, reflecting typological differences.

4.4 Results

Our main results are reported in Table 2.

We use FSM-constrained decoding as the default decoding method. Across all five treebanks, our lightweight models achieve competitive UAS/LAS under this constrained decoding setup.

The raw representation already reaches strong accuracy, and the piped representation further improves attachment accuracy, bringing performance closer to stronger baselines. When combined with consensus decoding, the piped model yields additional gains on the datasets we have completed so far.

In addition to accuracy, our approach is substantially more computationally efficient, in contrast to prior LLM-based approaches trained with significantly larger compute budgets (e.g., multi-A100 setups). This suggests that small but well-constrained models can provide a practical alternative for dependency parsing when training and deployment resources are limited.

Compared to the raw representation, the piped representation achieves more improvements in LAS. This behavior is likely related to the staged prediction strategy, where head indices are predicted first and relation labels are conditioned on the predicted heads. Such decomposition reduces the complexity of relation prediction and aligns with our design intuition that separating structure and labeling can benefit relation accuracy.

Model	UAS	LAS	Speed (sen/s)
bart-base (139M)	89.0	85.1	108.84
bart-large (406M)	94.1	90.5	42.71

Table 3: Model Size vs. Accuracy and Training Speed

5 Analysis

5.1 Model Size vs. Accuracy and Speed

Table 3 compares BART-base and BART-large under identical settings on `en_ewt`.

While BART-large achieves higher accuracy, it is significantly more expensive to train, with training speed reduced by more than half compared to BART-base. This result shows that increasing model size can improve parsing accuracy, but the gains quickly diminish as computational cost grows, especially under limited resources.

6 Conclusion

We presented a lightweight, structure-aware parsing framework that combines compact encoder–decoder models with finite-state constraints and piped decoding. Experiments on multiple UD treebanks demonstrate that carefully constrained small-scale models can approach the performance of much larger language models while using a fraction of the computational resources. These findings suggest that balancing accuracy and efficiency is crucial for practical parsing in low-resource or deployment-constrained scenarios, and that structural guidance can allow modest models to rival far more expensive alternatives.

[1] The code is available in the repository: https://github.com/yuanyunzhe/ldp.

[2] All experiments run on a single NVIDIA RTX 4060 Ti GPU with 16 GB memory.

7 Limitations

Our experiments are conducted on five UD treebanks and use a length-based filtering strategy ($\leq$ 60 tokens) due to GPU memory constraints. As a result, the reported scores may not fully reflect performance on longer sentences or on the full, unfiltered test sets. Extending evaluation to more languages and longer inputs is an important direction for future work.

We used an AI assistant to help with English phrasing and LaTeX formatting; all technical content, experimental results, and claims were verified by the authors.

References

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750.

Daniel Deutsch, Shyam Upadhyay, and Dan Roth. 2019. A general-purpose algorithm for constrained sequential inference. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 482–492.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. 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 4171–4186.

Timothy Dozat and Christopher D Manning. 2016. Deep biaffine attention for neural dependency parsing. *arXiv preprint arXiv:1611.01734*.

Saibo Geng, Martin Josifoski, Maxime Peyrard, and Robert West. 2023. Grammar-constrained decoding for structured nlp tasks without finetuning. *arXiv preprint arXiv:2305.13971*.

Han He and Jinho D Choi. 2023. Unleashing the true potential of sequence-to-sequence models for sequence tagging and structure parsing. *Transactions of the Association for Computational Linguistics*, 11:582–599.

Claudiu Daniel Hromei, Danilo Croce, and Roberto Basili. 2024. U-deppllama: Universal dependency parsing via auto-regressive large language models. *IJCoL. Italian Journal of Computational Linguistics*, 10(10, 1).

Richard Johansson and Pierre Nugues. 2008. Dependency-based syntactic–semantic analysis with propbank and nombank. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 183–187.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Dan Kondratyuk and Milan Straka. 2019. 75 languages, 1 model: Parsing universal dependencies universally. *arXiv preprint arXiv:1904.02099*.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, and Noah A Smith. 2016. Distilling an ensemble of greedy dependency parsers into one mst parser. *arXiv preprint arXiv:1609.07561*.

Hiroshi Matsuda, Chunpeng Ma, and Masayuki Asahara. 2025. Step-by-step instructions and a simple tabular output format improve the dependency parsing accuracy of llms. *arXiv preprint arXiv:2506.09983*.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Khalil Mrini, Franck Dernoncourt, Quan Tran, Trung Bui, Walter Chang, and Ndapa Nakashole. 2019. Rethinking self-attention: Towards interpretability in neural parsing. *arXiv preprint arXiv:1911.03875*.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of IWPT*.

Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*.

Kenji Sagae and Alon Lavie. 2006. Parser combination by reparsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 129–132.

Milan Straka. 2018. UDPipe 2.0 prototype at CoNLL 2018 UD shared task. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 197–207, Brussels, Belgium. Association for Computational Linguistics.

Yuanhe Tian, Yan Song, and Fei Xia. 2022. Enhancing structure-aware encoder with extremely limited data for graph-based dependency parsing. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 5438–5449.

Pavlo Vasylenko, Pere-Lluís Huguet Cabot, Abelardo Carlos Martínez Lorenzo, and Roberto Navigli. 2023. Incorporating graph information in transformer-based amr parsing. *arXiv preprint arXiv:2306.13467*.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT*, volume 3, pages 195–206. Nancy, France.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Junru Zhou and Hai Zhao. 2019. Head-driven phrase structure grammar parsing on penn treebank. *arXiv preprint arXiv:1907.02684*.