Hierarchical Bracketing Encodings Work for Dependency Graphs

Ana Ezquerro, Carlos Gómez-Rodríguez and David Vilares
Universidade da Coruña, CITIC
Departamento de Ciencias de la Computación y Tecnologías de la Información
Campus de Elviña s/n, 15071
A Coruña, Spain
{ana.ezquerro, carlos.gomez, david.vilares}@udc.es

Abstract

We revisit hierarchical bracketing encodings from a practical perspective in the context of dependency graph parsing. The approach encodes graphs as sequences, enabling linear-time parsing with n tagging actions, and still representing reentrancies, cycles, and empty nodes. Compared to existing graph linearizations, this representation substantially reduces the label space while preserving structural information. We evaluate it on a multilingual and multi-formalism benchmark, showing competitive results and consistent improvements over other methods in exact match accuracy.

1 Introduction

Sequence labeling (SL) offers a simple yet effective paradigm for a wide range of natural language problems. By assigning one label to each token, sequence labeling models eliminate the need for complex decoding algorithms, and make inference simpler, faster and scalable in structured prediction settings. When paired with neural encoders (Ma and Hovy, 2016)—especially pre-trained encoders (Devlin et al., 2019)—SL models achieve strong results with minimal architectural complexity.

Recently, sequence labeling has been used to address various flavors of syntactic *tree* parsing such as continuous (Gómez-Rodríguez and Vilares, 2018; Kitaev and Klein, 2020; Amini and Cotterell, 2022) and discontinuous constituency parsing (Vilares and Gómez-Rodríguez, 2020), and dependency parsing (Strzyz et al., 2019; Vacareanu et al., 2020; Amini et al., 2023; Gómez-Rodríguez et al., 2023). Some approaches rely on positional offsets to indicate explicit relations between tokens, others use explicit bracketing schemes, and some derive from transition-based parsing systems. Despite this progress, extending SL to graph parsing introduces new challenges. Graphs may include reentrancies, cycles, and disconnected components—structures that cannot be directly captured by most tree-based linearizations. As a result, most graph parsers rely on graph- (Dozat and Manning, 2018; Wang et al., 2019) or transition-based (Fernández-González and Gómez-Rodríguez, 2020) decoders. To date, only Ezquerro et al. (2024) have shown that SL can be effectively adapted to this setting, suggesting it may offer a viable alternative that simplifies decoding while keeping the linearizations learnable.

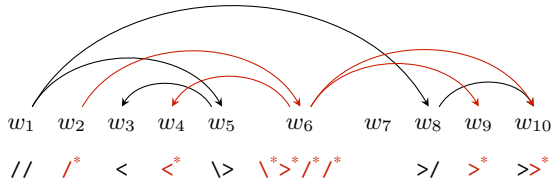
Our work builds upon recent efforts to encode edge information through bracketing schemes, previously applied to dependency trees (Strzyz et al., 2019) and graphs (Ezquerro et al., 2024). Namely, we apply the optimal hierarchical bracketing encoding from Ezquerro et al. (2025), a framework theoretically defined for both trees and graphs but empirically validated only on dependency trees, and provide the first empirical evaluation on dependency graphs. We evaluate its non-projective variant on a large benchmark of graph annotations, where re-entrancies and cycles are possible. Experiments show that this encoding preserves the performance of the original bracketing encodings of Ezquerro et al. (2024), while achieving a reduced label space and maintaining full coverage of dependency graphs. Notably, it improves exact match scores, likely due to a more balanced and compact label distribution.

2 Bracketing encodings for graphs

Let $W = (w_1, ..., w_n) \in \mathcal{V}^n$ be an input sentence from a vocabulary $\mathcal{V}$. A dependency graph built upon W is defined as $G = (W, A)$, where each element of A is an arc $(h \rightarrow d)$, such that $h \in [0, n]$ and $d \in [1, n]$, that connects a dependent (w_d) with its head (w_h). Note that, unlike in dependency trees, A is not constrained by connectivity and acyclicity properties. Following Ezquerro et al. (2024), we define an SL framework as an encoding

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 8838–8851

(a) Bracketing encoding with $k = 2$.



(b) Hierarchical bracketing encoding.

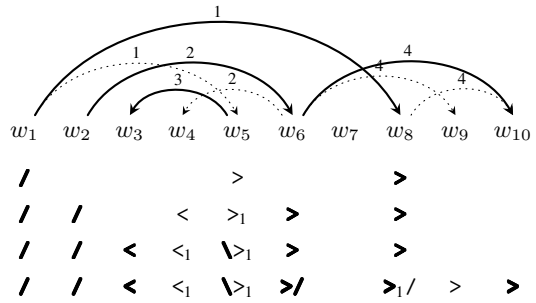


Figure 1: Bracketing and hierarchical bracketing encoding for the same dependency graph. In Figure 1a, crossing arcs in the same direction and their associated brackets are colored in red. In Figure 1b, structural arcs of the rope cover and their symbols are shown in bold, while auxiliary arcs are in dotted lines. Arc labels identify the structural set they belong to, while bracket subindices are used by the encoding to support crossing arcs. Each row shows the brackets added when encoding each structural set.

($\varepsilon : \mathcal{A}^n \rightarrow \mathcal{L}^n$)[1] and decoding ($\delta : \mathcal{L}^n \rightarrow \mathcal{A}^n$) function that represent a graph's arc information as a label sequence of length n over a label set $\mathcal{L}$.

Ezquerro et al. (2024) adapted Strzyz et al. (2019)'s bracketing encoding for dependency trees to graphs. In the bracketing encoding, each arc is represented with two balanced brackets that are included in the labels of its head and dependent: (<,\) for left arcs and (/,>) for right arcs. The decoding process uses a left-to-right pass that reads the sequence of brackets, pushing positions with opening brackets into each stack and popping elements when a closing bracket is found. To solve the limitation of crossing arcs in the same direction (see Figure 1a), they proposed distributing the arcs of A in k *relaxed planes*, where each plane does not contain crossing arcs in the same direction; and encode each plane with different bracket symbols, so only brackets from the same plane match each other at decoding time.

Although Ezquerro et al. (2024) achieved strong performance on a large multilingual benchmark using $k \leq 3$, the bracketing encoding presents key limitations in terms of its theoretical coverage[2]. The hyperparameter k inherently constrains the representation to graph structures with no more than k relaxed planes. While Ezquerro et al. (2024) explored increasing this value, doing so comes at the computational cost of expanding the label set $\mathcal{L}$. Their bounded bit-based encodings ($4k$ and $6k$-bit) address this issue and fix the cardinality of the label set, but they still rely on a fixed k, thereby maintaining the same limitation on theoretical coverage.

We fill this gap by operationalizing the concept of rope covers (Yli-Jyrä, 2019) within the hierarchical bracketing encoding, defined theoretically in Ezquerro et al. (2025), for graphs, to successfully apply a more compressed bracketing representation that removes the need for an hyperparameter k.

3 Hierarchical Bracketing Encoding

Given a graph $G = (W, A)$, a *rope cover* is a subset $R \subseteq A$ such that every arc in $A \setminus R$ leans[3] on at least one arc of R; and it is said to be *proper* if no arc in R leans on another arc of R. The arcs of R as denoted as *structural* arcs, and the arcs of $A \setminus R$ as *auxiliary* arcs. Yli-Jyrä (2019) demonstrated that the proper rope cover is unique and defined an algorithm to find it for any arbitrary graph; and Ezquerro et al. (2025) showed that it is optimal in the number of structural arcs. Extending this terminology, a proper rope cover R gives rise to the concept of a structural set: a maximal subset $S \subseteq A$ that contains one arc from R, while all other arcs in S lean on it.

Figure 1b shows the proper rope cover for a dependency graph. See that the first structural arc is $(1 \rightarrow 8)$ and the only arc that leans on it is $(1 \rightarrow 5)$, so these two arcs together form a structural set. The arc $(6 \rightarrow 10)$ is also a structural arc and its auxiliary arcs are $(6 \rightarrow 9)$ and $(8 \rightarrow 10)$, and the set of these three arcs is also a structural set.

Encoding and decoding structural sets Ezquerro et al. (2025) proposed an algorithm to independently encode each structural set in an optimal number of unique labels, using *balanced super-*

[1]Here we use $\mathcal{A}^n$ to denote all the possible sets of arcs for a graph of size n.

[2]Theoretical coverage is defined as the ratio of graphs in a reference treebank for which the linearization algorithm can produce a lossless encoding–decoding cycle, i.e., the original graph can be perfectly reconstructed from its encoded form.

[3]We say that an arc $(h \rightarrow d)$ leans on another $(h' \rightarrow d')$ if $(h' \rightarrow d')$ covers $(h \rightarrow d)$ and either $\min(h, d) = \min(h', d')$ or $\max(h, d) = \max(h', d')$.

brackets for structural arcs (****, **>**, **<**, **/**) and one bracket symbol (\, >, <, /) to encode the non-leant position of each auxiliary arc. See Figure 1b (row 1): the first structural set is $S_1 = \{(1 \rightarrow 8), (1 \rightarrow 5)\}$ and $(1 \rightarrow 8)$ is the structural arc, which is encoded with balanced superbrackets (**/**,**>**). The auxiliary arc $(1 \rightarrow 5)$ is encoded with only one bracket (>) on the non-leant position (in this case, 1 is the leant position, so the bracket corresponding to position 5 is the one encoded).

The decoding process reads the bracket sequence from left to right, pushing any opening symbol (**/**, /, **<**, <) into a stack. When a closing superbracket is found (****, **>**), the system pops stack elements, matching brackets with said superbracket, until an opening superbracket is found (**<**, **/**). Instead, when a closing bracket is found (\, >), the system matches the opening superbracket that should be on top of the stack with the closing bracket.

Encoding and decoding crossing arcs When jointly encoding different structural sets with crossing arcs, the decoding previously defined leads to incorrect arcs[4]. For instance, if we consider only the symbols in Figure 1b (row 2): (**/**, **/**, ·, <, >, **>**, ·, **>**, ·, ·), the decoding algorithm recovers $(2 \rightarrow 5)$ instead of $(1 \rightarrow 5)$ since the opening superbracket (**/**) of the structural arc $(2 \rightarrow 6)$ is found first in the decoding stack. To solve this issue, we adopt the indexing approach from Ezquerro et al. (2025) and add an index in those symbols that require skipping matching structural arcs at decoding time. In Figure 1b (row 2), when adding the index 1 to w_5's bracket ($>_1$), the decoding system skips one structural arc when parsing the stack, which results in matching the bracket $>_1$ with the structural arc in position 1, resolving the auxiliary arc $(1 \rightarrow 5)$. When adding the structural arc $(5 \rightarrow 3)$, w_4's bracket index needs to be increased so it resolves to $(6 \rightarrow 4)$ rather than $(5 \rightarrow 4)$.

Postprocessing As is common in parsing as sequence labeling, the mapping from graphs to label sequences is not surjective. Thus, when implementing this approach in practice by training a sequence labeling system to produce label sequences, the predicted sequences are not guaranteed to be structurally valid. To prevent decoding errors, such as popping an element from an empty stack, we apply a separate postprocessing step to correct the sequence and ensure a well-formed dependency graph. Specifically, this process matches any unbalanced closing brackets with the dummy start node (w_0) and discards any unclosed superbrackets left on the stack at the end of decoding.

[4]Note that auxiliary arcs from the same structural set cannot cross each other in the same direction; and when they cross in different directions, they can still be recovered since the bracket direction already determines the superbracket that matches each arc.

4 Experiments

The main goal of this work is to assess whether hierarchical bracketing encodings can effectively support dependency graph parsing within a neural sequence labeling framework. Our source code to reproduce our experiments is available at `https://github.com/anaezquerro/separ`.

Neural tagger For our experiments, we reproduce the same tagger as in Ezquerro et al. (2024) to learn the encoding labels and the arc-relation module to learn arc labels from the hidden representations of the predicted arcs. Specifically, we rely on XLM-RoBERTa (Conneau et al., 2020) for non-English and XLNet (Yang et al., 2019) for English treebanks as neural encoders, and two 1-layered FFNs for label and relation prediction. For an input sentence $W = (w_1, .., w_n) \in \mathcal{V}^n$, our encoder computes its contextualized embeddings $(\mathbf{h}_1, ..., \mathbf{h}_n) \in \mathbb{R}^{n \times D}$ and uses the first FFN to learn the label distribution for a specific encoding. For inference, the sequence of predicted labels is used to run the decoding process and recover a set of predicted arcs, denoted as $\hat{A} = \{(h \rightarrow d) : h \in [0, n], d \in [1, n]\}$. To predict the arc relation of each predicted arc, the hidden representations of its head and dependent are concatenated and fed to the second FFN, which learns the arc-relation distribution from these combined representations. To optimize the weights of this second FFN, our framework uses the gold set of arcs at training time to concatenate head and dependent representations. The full architecture is optimized using the cross-entropy losses of both FFNs.

Datasets We used five datasets from SemEval 2015 Task 18 (Oepen et al., 2015) with semantic annotations: (i) the English dataset annotated with DELPH-IN MRS-Derived Bi-Lexical Dependencies (DM Ivanova et al., 2012), (ii) the English and Chinese datasets with Enju Predicate-Argument Structures (PAS Miyao et al., 2005), and (iii) the English and Czech datasets with Prague Semantic Dependencies (PSD Hajič et al., 2012); and the

	$\mathbf{B_2}$		$\mathbf{B_3}$		$\mathbf{B6_3}$		$\mathbf{B6_4}$		**HB**		**Biaf.**	
	LF	LM	LF	LM	LF	LM	LF	LM	LF	LM	LF	LM
en$_{\text{DM}}$	94.57_{100}	52.27_{100}	94.67_{100}	52.48_{100}	$94.04_{99.51}$	$43.69_{83.62}$	$\mathbf{94.74}_{99.96}$	$51.77_{98.51}$	94.52	**53.05**	95.45	46.95
en$_{\text{PAS}}$	95.27_{*100}	$47.02_{99.93}$	95.50_{100}	49.65_{100}	$93.56_{97.56}$	$24.47_{42.55}$	$95.01_{99.37}$	$42.27_{78.30}$	**95.59**	**51.84**	96.13	48.51
en$_{\text{PSD}}$	$85.33_{99.96}$	$15.82_{98.58}$	85.95_{*100}	$17.09_{99.86}$	$85.68_{99.98}$	$15.67_{98.94}$	$\mathbf{86.20}_{*100}$	$\mathbf{17.66}_{99.93}$	85.61	16.67	86.84	16.60
cs$_{\text{PSD}}$	$90.02_{99.96}$	$28.80_{98.02}$	90.03_{100}	29.40_{100}	$90.25_{99.98}$	$29.52_{99.22}$	$\mathbf{90.48}_{*100}$	$\mathbf{31.56}_{99.82}$	89.64	30.84	91.21	27.90
zh$_{\text{PAS}}$	87.20_{*100}	$31.72_{99.77}$	$\mathbf{88.67}_{100}$	32.70_{100}	$84.67_{96.57}$	$21.41_{46.79}$	$87.72_{98.30}$	$27.18_{71.40}$	88.09	**34.35**	90.38	34.06
ar$_{\text{PADT}}$	$83.13_{99.97}$	$11.47_{98.09}$	82.75_{*100}	$10.15_{99.85}$	$82.72_{99.97}$	$11.62_{98.68}$	$\mathbf{83.17}_{99.98}$	$\mathbf{13.24}_{99.56}$	82.54	11.47	85.26	11.03
bg$_{\text{BTB}}$	93.97_{100}	47.76_{100}	92.89_{100}	47.94_{100}	$\mathbf{94.29}_{99.99}$	$\mathbf{51.52}_{99.64}$	$93.32_{99.99}$	$50.45_{99.73}$	92.83	50.18	94.92	48.75
fi$_{\text{TDT}}$	$90.35_{99.93}$	$42.57_{97.94}$	90.50_{*100}	$43.47_{99.87}$	$\mathbf{91.03}_{99.93}$	$\mathbf{46.24}_{98.39}$	$90.81_{99.97}$	$45.47_{99.16}$	90.10	45.59	92.33	45.79
fr$_{\text{SEQ.}}$	92.98_{*100}	$37.72_{99.56}$	92.47_{100}	38.16_{100}	$\mathbf{93.98}_{99.97}$	$45.18_{98.46}$	93.88_{100}	$\mathbf{47.81}_{100}$	92.97	44.96	94.91	47.59
it$_{\text{ISDT}}$	93.16_{*100}	$45.44_{99.79}$	93.47_{100}	45.02_{100}	93.51_{*100}	$47.93_{99.59}$	93.52_{100}	48.76_{100}	**93.75**	**49.38**	94.36	48.13
lt$_{\text{ALK.}}$	$83.50_{99.92}$	$18.57_{97.22}$	$80.79_{99.99}$	$19.74_{99.56}$	$\mathbf{84.85}_{99.97}$	$21.93_{98.68}$	$81.75_{99.99}$	$\mathbf{22.37}_{99.71}$	79.12	20.32	83.82	20.47
lv$_{\text{LVTB}}$	$87.80_{99.95}$	$37.47_{98.41}$	$87.12_{99.99}$	$35.27_{99.78}$	$87.78_{99.94}$	$38.29_{98.03}$	$\mathbf{88.30}_{99.98}$	$\mathbf{39.99}_{99.23}$	87.46	38.51	89.84	39.66
pl$_{\text{PDB}}$	$93.25_{99.97}$	$51.33_{98.69}$	93.61_{*100}	$53.41_{99.82}$	$\mathbf{93.63}_{99.97}$	$\mathbf{53.72}_{98.92}$	$93.61_{99.99}$	$53.27_{99.50}$	92.60	52.46	94.40	52.55
ru$_{\text{SYN.}}$	$93.74_{99.99}$	$53.12_{99.65}$	93.64_{100}	52.47_{100}	93.86_{*100}	$54.01_{99.85}$	$\mathbf{93.93}_{*100}$	$54.00_{99.95}$	93.54	**54.24**	94.45	51.93
sk$_{\text{SNK}}$	$92.08_{99.99}$	$50.90_{99.72}$	91.83_{100}	49.29_{100}	$\mathbf{92.51}_{99.99}$	$52.87_{99.62}$	92.41_{*100}	$53.16_{99.91}$	92.11	**54.85**	93.99	54.38
sv$_{\text{TAL.}}$	89.90_{*100}	$38.64_{99.84}$	89.50_{100}	38.23_{100}	$\mathbf{91.12}_{99.98}$	$\mathbf{45.28}_{99.34}$	90.97_{*100}	$43.23_{99.92}$	89.87	42.00	92.05	41.43
ta$_{\text{TTB}}$	61.89_{100}	1.67_{100}	61.89_{100}	1.67_{100}	$\mathbf{66.08}_{100}$	2.50_{100}	66.08_{100}	2.50_{100}	63.67	**3.33**	64.83	2.50
uk$_{\text{IU}}$	$89.94_{99.99}$	$34.30_{99.10}$	89.96_{100}	32.74_{100}	$90.38_{99.99}$	$37.33_{99.10}$	$\mathbf{90.59}_{*100}$	$\mathbf{37.89}_{99.78}$	89.50	37.44	92.17	38.34
μ	88.85	36.13	88.73	36.34	89.19	36.24	**89.30**	38.31	88.59	**38.50**	90.45	37.69

Table 1: LF and LM performance for the bracketing (**B**), $6k$-bit (**B6**) and hierarchical bracketing (**HB**, ours) encodings, and the biaffine parser (**Biaf.**). In the acronyms, subscripts indicate the value of k, whereas in the scores, they denote the coverage. The coverage of **HB** and **Biaf.** is not indicated because it is always guaranteed to be 100%. Best SL approach is highlighted in bold, and the best parser is underlined. The English and Czech results correspond to the in-distribution set. The last row (μ) shows the average across all treebanks.

IWPT 2021 Shared Task datasets (Bouma et al., 2021) with enhanced dependencies in 17 diverse languages[5].

Evaluation We use the SDP evaluation toolkit[6] (Oepen et al., 2015) to report both the labeled F1 score and exact match (LF, LM). Results with further metrics are included in the Appendix A.3.

Baselines We run our tagging model with the bracketing and $6k$-bit encodings from Ezquerro et al. (2024). For external assessment, we also report the performance of the biaffine parser (Dozat and Manning, 2018)—a graph-based, non-sequence-labeling model—providing a consistent reference of the state of the art in graph parsing.

5 Results

Table 1 shows the performance of different graph encodings and the biaffine baseline. We use acronyms to refer to the bracketing (B) and $6k$-bit (B6) encodings from Ezquerro et al. (2024) and subindices to specify the value of the hyperparameter k. In terms of LF, our hierarchical bracketing encoding (HB) performs competitively, with an average that is only 0.14 below B_3, 0.26 below B_2, 0.60 below $B6_3$, and 0.71 below the best-performing encoding, $B6_4$. Still, a more fine-grained analysis shows that HB outperforms other graph encodings in only two treebanks (English-PAS and Italian-ISDT), although it is still surpassed by the biaffine baseline in both cases. Overall, biaffine obtains the highest LF score (90.45), followed by $B6_4$ (89.30).

The outcome shifts when focusing on the exact match performance. Our HB outperforms other approaches, including the biaffine baseline, in 7 treebanks and obtains the best LM score on average (HB: 38.50), followed by the $6k$-bit encoding ($B6_4$: 38.31). When comparing coverage and performance between $B6_4$ and HB (coverage data are in Table 1 and Tables 5-21 in the Appendix), we observe that $B6_4$'s LM score lags behind in the datasets where its coverage is most limited at $k = 4$, like the DM and PAS treebanks, whereas HB handles them better with its guaranteed 100% coverage.

We also compared the label space of our HB with Ezquerro et al. (2024)'s approaches. Figure 2a

[5]We report metrics on the largest treebank of each language, excluding the English and Czech languages since they are already included in the SemEval 2015 Task 18 dataset.

[6]https://github.com/semantic-dependency-parsing/toolkit.

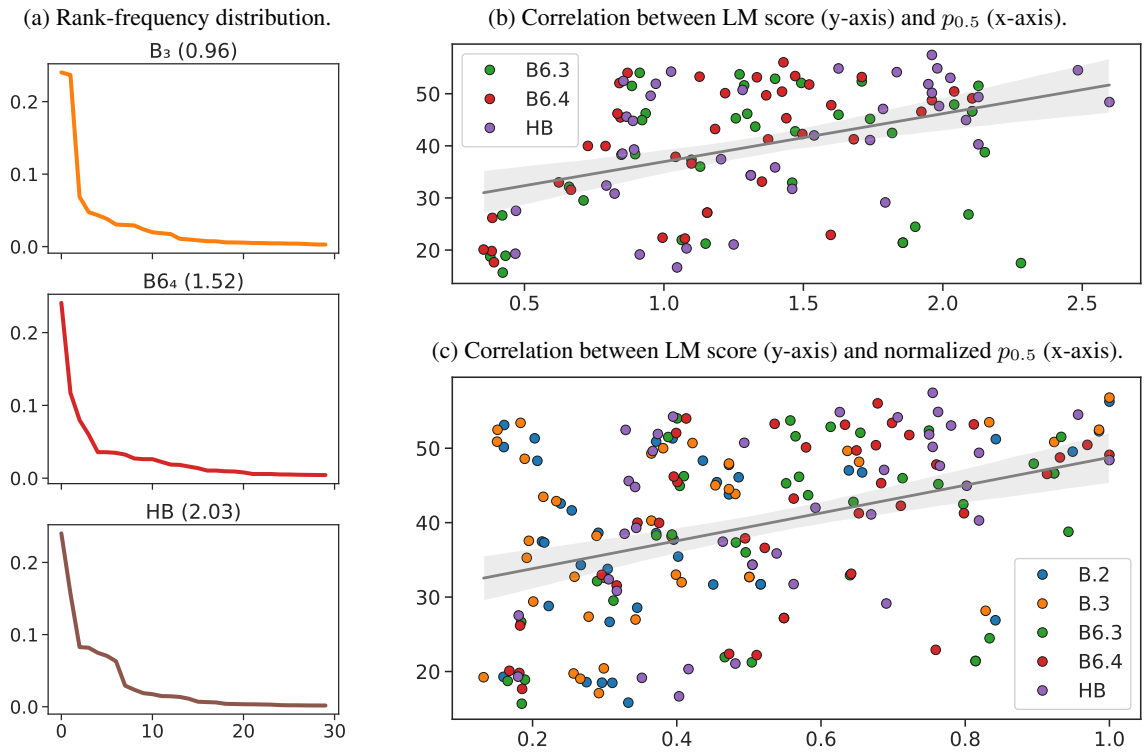


Figure 2: Analysis of the correlation between the encoding performance and the label distribution. Figure 2a shows the rank-frequency distribution of labels in the English-DM. Figures 2b and 2c visualize the correlation between the LM score and $p_{0.5}$ across all treebanks.

shows the rank-frequency distribution of different encoding labels, with a numeric value that indicates the relative rank at the 50th percentile ($p_{0.5}$). For instance, 2.03% of the most frequent labels of HB account for approximately half of all occurrences. Instead, in B_3, 0.96% of the most frequent labels represent half of the occurrences. These values measure the balance of different encodings, in terms of which ratio of the most frequent labels concentrate 50% of the probability mass. We contrast this value against the LM score in Figure 2b, where a significant linear correlation of 88.1% with a p-value < 0.001 was detected, indicating a strong dependency between balanced label distributions and exact match performance. We repeated this test adding the scores of the bracketing encodings[7] but normalizing the values of the $p_{0.5}$ and obtained again a significant correlation of 83.6% (Figure 2c).

6 Conclusion

In this work we apply the non-projective hierarchical bracketing encoding of Ezquerro et al. (2025) to graphs instead of trees. The method offers a competitive trade-off between coverage, label space compactness and parsing performance. Although it does not yield the highest LF score, it achieves the best average exact match across a large multilingual and multi-formalism benchmark. We further showed that the method yields a more balanced label distribution, which correlates strongly with the exact match performance in graph encodings. To our knowledge, this is the first work to empirically demonstrate that hierarchical bracketing encodings can be effectively learned and applied to dependency graph parsing.

[7]We excluded the bracketing scores from Figure 2b since their $p_{0.5}$ follows a different distribution.

Acknowledgments

We acknowledge grants GAP (PID2022-139308OA-I00) funded by MICIU/AEI/10.13039/501100011033/ and ERDF, EU; LATCHING (PID2023-147129OB-C21) funded by MICIU/AEI/10.13039/501100011033 and ERDF, EU; and TSI-100925-2023-1 funded by Ministry for Digital Transformation and Civil Service and "NextGenerationEU" PRTR; as well as funding by Xunta de Galicia (ED431C 2024/02), and CITIC, as a center accredited for excellence

within the Galician University System and a member of the CIGUS Network, receives subsidies from the Department of Education, Science, Universities, and Vocational Training of the Xunta de Galicia. Additionally, it is co-financed by the EU through the FEDER Galicia 2021-27 operational program (Ref. ED431G 2023/01).

Limitations

Graph formalisms While a wide range of graph-based annotations have been proposed across NLP tasks – such as structured sentiment analysis, emotion-cause analysis, and other forms of relational inference – we focus our experimental study on two well-established formalisms: semantic graph parsing and enhanced dependency parsing. These frameworks offer clear benchmarks and annotation standards, allowing us to evaluate our models in a controlled and widely studied setting.

Computational resources Our experiments were conducted using local high-performance computing infrastructure, with full access to 8 NVIDIA RTX 3090 GPUs (24GB each) and 3 NVIDIA RTX 6000 GPUs (48GB each). These resources allowed us to efficiently train and evaluate all models presented in this work.

Unboundedness Our proposed encoding is unbounded, meaning that the number of labels needed to encode a given dataset or tree is not theoretically bounded by a constant. This is a purely theoretical limitation, but our experiments show that it is not relevant in practice for the dataset tested, as in practice it requires fewer labels than the bounded encodings $B6_3$ and $B6_4$ (Table 4 in the Appendix).

Ethical considerations

Our work benchmarks the hierarchical bracketing encoding proposed by Ezquerro et al. (2025) in the context of dependency graph parsing. None of the materials used involve sensitive personal data, human subjects, or contexts that might pose ethical risks. Consequently, the techniques can be incorporated into research without ethical reservations.

We acknowledge the environmental impact of training neural models, particularly with respect to CO_2 emissions. All experiments were conducted in Spain, where we measured the carbon footprint during both training and inference. Training produces around 0.28 g CO_2 per epoch and inference approximately 0.19 g CO_2. These values remain low compared to recent large NLP models. For comparison, the European Union sets a limit of roughly 115 g CO_2 per kilometer for newly manufactured cars.

References

Afra Amini and Ryan Cotterell. 2022. On parsing as tagging. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8884–8900, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Afra Amini, Tianyu Liu, and Ryan Cotterell. 2023. Hexatagging: Projective dependency parsing as tagging. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1453–1464, Toronto, Canada. Association for Computational Linguistics.

Gosse Bouma, Djamé Seddah, and Daniel Zeman. 2021. From Raw Text to Enhanced Universal Dependencies: The Parsing Shared Task at IWPT 2021. In *Proceedings of the 17th International Conference on Parsing Technologies and the IWPT 2021 Shared Task on Parsing into Enhanced Universal Dependencies (IWPT 2021)*, pages 146–157, Online. 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.

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, Minneapolis, Minnesota. Association for Computational Linguistics.

Timothy Dozat and Christopher D. Manning. 2018. Simpler but More Accurate Semantic Dependency Parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 484–490, Melbourne, Australia. Association for Computational Linguistics.

Ana Ezquerro, David Vilares, and Carlos Gómez-Rodríguez. 2024. Dependency Graph Parsing as Sequence Labeling. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11804–11828, Miami, Florida, USA. Association for Computational Linguistics.

Ana Ezquerro, David Vilares, Anssi Yli-Jyrä, and Carlos Gómez-Rodríguez. 2025. Hierarchical bracketing encodings for dependency parsing as tagging. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 18436–18450, Vienna, Austria. Association for Computational Linguistics.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2020. Transition-based Semantic Dependency Parsing with Pointer Networks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7035–7046, Online. Association for Computational Linguistics.

Carlos Gómez-Rodríguez, Diego Roca, and David Vilares. 2023. 4 and 7-bit labeling for projective and non-projective dependency trees. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6375–6384, Singapore. Association for Computational Linguistics.

Carlos Gómez-Rodríguez and David Vilares. 2018. Constituent Parsing as Sequence Labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1314–1324, Brussels, Belgium. Association for Computational Linguistics.

Jan Hajič, Eva Hajičová, Jarmila Panevová, Petr Sgall, Ondřej Bojar, Silvie Cinková, Eva Fučíková, Marie Mikulová, Petr Pajas, Jan Popelka, Jiří Semecký, Jana Šindlerová, Jan Štěpánek, Josef Toman, Zdeňka Urešová, and Zdeněk Žabokrtský. 2012. Announcing Prague Czech-English Dependency Treebank 2.0. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 3153–3160, Istanbul, Turkey. European Language Resources Association (ELRA).

Angelina Ivanova, Stephan Oepen, Lilja Øvrelid, and Dan Flickinger. 2012. Who Did What to Whom? A Contrastive Study of Syntacto-Semantic Dependencies. In *Proceedings of the Sixth Linguistic Annotation Workshop*, pages 2–11, Jeju, Republic of Korea. Association for Computational Linguistics.

Nikita Kitaev and Dan Klein. 2020. Tetra-Tagging: Word-Synchronous Parsing with Linear-Time Inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6255–6261, Online. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. *Preprint*, arXiv:1711.05101.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Yusuke Miyao, Takashi Ninomiya, and Jun'ichi Tsujii. 2005. Corpus-Oriented Grammar Development for Acquiring a Head-Driven Phrase Structure Grammar from the Penn Treebank. In *Natural Language Processing – IJCNLP 2004*, pages 684–693, Berlin, Heidelberg. Springer Berlin Heidelberg.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajič, and Zdeňka Urešová. 2015. SemEval 2015 Task 18: Broad-Coverage Semantic Dependency Parsing. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 915–926, Denver, Colorado. Association for Computational Linguistics.

Michalina Strzyz, David Vilares, and Carlos Gómez-Rodríguez. 2019. Viable Dependency Parsing as Sequence Labeling. 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 717–723, Minneapolis, Minnesota. Association for Computational Linguistics.

Robert Vacareanu, George Caique Gouveia Barbosa, Marco A. Valenzuela-Escárcega, and Mihai Surdeanu. 2020. Parsing as tagging. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 5225–5231, Marseille, France. European Language Resources Association.

David Vilares and Carlos Gómez-Rodríguez. 2020. Discontinuous Constituent Parsing as Sequence Labeling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2771–2785, Online. Association for Computational Linguistics.

Xinyu Wang, Jingxian Huang, and Kewei Tu. 2019. Second-Order Semantic Dependency Parsing with End-to-End Neural Networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4609–4618, Florence, Italy. Association for Computational Linguistics.

Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. 2015. Empirical Evaluation of Rectified Activations in Convolutional Network. *Preprint*, arXiv:1505.00853.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Anssi Yli-Jyrä. 2019. Transition-Based Coding and Formal Language Theory for Ordered Digraphs. In *Proceedings of the 14th International Conference on Finite-State Methods and Natural Language Processing*, pages 118–131, Dresden, Germany. Association for Computational Linguistics.

A Appendix

A.1 Treebank statistics

This work conducts experiments in all treebanks of the SemEval 2015 Task 18 dataset (Oepen et al., 2015) and a selection of IWPT datasets (Bouma et al., 2021). Tables 2 and 3 summarize different statistics of each treebank. Table 4 shows information about the labels generated in each encoding.

	#sents	n	**%r.planes** 1	2	3	**d.**	$\|R\|$	**#cycs**
en_{DM}	33964	22.52	86.28	13.68	0.05	0.78	9.87	0
	1692	22.28	86.94	13.06	0.00	0.79	9.79	0
	1410	22.66	84.11	15.89	0.00	0.77	9.80	0
	1849	17.08	88.64	11.20	0.16	0.75	7.26	0
en_{PAS}	33964	22.52	83.70	16.20	0.10	1.01	10.95	0
	1692	22.28	86.52	13.42	0.06	1.00	10.86	0
	1410	22.66	82.91	17.02	0.07	1.01	10.92	0
	1849	17.08	83.02	16.77	0.22	0.99	8.35	0
en_{PSD}	33964	22.52	77.51	20.99	1.44	0.70	7.89	0
	1692	22.28	76.71	22.16	0.95	0.70	7.84	0
	1410	22.66	77.38	21.21	1.28	0.69	7.81	0
	1849	17.08	81.23	17.79	0.87	0.67	5.59	0
cs_{PSD}	40047	23.45	74.22	24.04	1.66	0.77	8.90	0
	2010	22.99	75.32	23.33	1.24	0.78	8.82	0
	1670	22.99	73.35	24.67	1.98	0.76	8.53	0
	5226	16.82	78.66	19.15	2.01	0.78	6.31	0
zh_{PAS}	25896	22.43	75.60	24.12	0.28	1.02	11.48	0
	2440	27.95	73.16	26.60	0.25	1.02	14.63	0
	8976	23.89	75.12	24.64	0.23	1.02	12.26	0

Table 2: Treebank statistics for the SDP datasets. Number of sentences (**#sents**), average sentence length (n), distribution of sentences by number of relaxed planes (**%r.planes**), density (**d**) as the average ratio of arcs and nodes per graph, average number of structural arcs ($|R|$) and number of cycles (**#cycs.**). Different splits in each subrow: train, development, ID and OOD or test.

A.2 Training configuration

All our models were trained using the AdamW optimizer (Loshchilov and Hutter, 2019) with a learning rate of $\eta = 10^{-5}$, for 100 epochs and with token-based batching set to 500 tokens. The FFN layers use the LeakyReLU activation (Xu et al., 2015) with a negative slope of 0.1 and a dropout rate set to 0.1. The evaluation on the development set was used as the stopping criterion.

A.3 Additional results

Tables 5 to 21 show the detailed performance of each encoding in the evaluation sets. We use a similar notation to Table 1, including the (un)labeled F1-score (***F**) and exact match (***M**), and the tag accuracy (**A**) and ratio of well-formed graphs (**W**).

	#sents	n	**%r.planes** 1	2	3	**d.**	$\|R\|$	**#cycs**
ar_{PADT}	6075	36.85	65.32	32.49	1.86	1.53	18.95	1783
	909	33.27	69.97	28.49	1.32	1.05	17.05	282
	680	41.56	64.26	33.82	1.76	1.05	21.31	235
bg_{BTB}	8907	13.96	90.14	9.68	0.18	1.02	6.41	1177
	1115	14.43	89.69	10.22	0.09	1.02	6.63	153
	1116	14.09	90.59	9.41	0.00	1.02	6.44	145
fi_{TDT}	12217	13.33	78.90	19.19	1.77	1.06	6.29	2906
	1364	13.42	78.30	20.01	1.32	1.06	6.31	340
	1555	13.55	77.17	20.77	1.93	1.07	6.34	357
$fr_{\text{SEQ.}}$	2231	22.64	80.55	19.23	0.18	1.04	9.45	770
	412	24.28	80.83	18.93	0.24	1.04	10.00	166
	456	22.04	79.82	19.74	0.44	1.04	9.05	119
it_{ISDT}	13121	21.04	85.73	14.17	0.10	1.03	9.19	2963
	564	21.11	87.06	12.94	0.00	1.03	9.27	136
	482	21.61	86.31	13.49	0.21	1.03	9.40	109
$lt_{\text{ALK.}}$	2341	20.35	51.82	44.55	3.03	1.10	9.77	388
	617	18.74	58.67	39.87	1.46	1.07	9.11	89
	684	15.86	53.22	44.01	2.34	1.08	7.42	63
lv_{LVTB}	10156	16.50	73.50	24.25	2.10	1.06	7.60	2189
	1664	15.60	75.12	22.84	1.86	1.04	7.13	255
	1823	14.48	78.72	19.69	1.37	1.02	6.66	297
$nl_{\text{ALP.}}$	12264	15.16	82.89	16.47	0.63	1.03	6.37	1575
	718	16.07	87.47	11.84	0.70	1.02	6.78	62
	596	18.53	76.17	22.99	0.84	1.04	7.68	68
pl_{PDB}	17722	15.90	67.57	30.98	1.37	1.06	7.55	2677
	2215	15.66	68.04	30.61	1.35	1.06	7.46	322
	2215	15.18	68.53	30.16	1.13	1.06	7.27	351
$ru_{\text{SYN.}}$	48814	17.83	67.29	32.30	0.41	1.04	8.41	4560
	6584	18.00	65.31	34.23	0.44	1.05	8.37	580
	6491	18.08	65.23	34.42	0.35	1.05	8.52	588
sk_{SNK}	8483	9.50	78.52	21.10	0.38	1.04	4.37	565
	1060	12.01	81.13	18.40	0.47	1.05	5.80	120
	1061	12.00	77.38	22.34	0.28	1.05	5.78	164
$sv_{\text{TAL.}}$	4303	15.49	85.99	13.97	0.05	1.05	6.52	812
	504	19.44	76.19	23.61	0.20	1.06	7.80	138
	1219	16.72	85.23	14.60	0.16	1.05	6.95	242
ta_{TBT}	400	15.82	97.75	2.25	0.00	1.02	8.21	1
	80	15.79	98.75	1.25	0.00	1.05	8.29	25
	120	16.57	98.33	1.67	0.00	1.03	8.41	43
uk_{IU}	5496	16.81	63.36	35.94	0.69	1.06	7.71	968
	672	18.71	61.61	37.80	0.60	1.07	8.83	198
	892	19.19	65.25	33.86	0.90	1.05	9.07	152

Table 3: Treebank statistics for the IWPT datasets. The notation is the same as in Table 2.

	$\mathbf{B_2}$		$\mathbf{B_3}$		$\mathbf{B6_3}$		$\mathbf{B6_4}$		**HB**	
en_{DM}	466	21	483	23	486	25	621	29	319	8
en_{PAS}	900	42	929	45	543	19	905	44	356	12
en_{PSD}	1013	51	1265	87	681	39	823	51	650	57
cs_{PSD}	1638	140	2021	204	856	67	1047	92	918	80
zh_{PAS}	1070	184	1117	198	474	36	774	85	377	49
ar_{PADT}	830	81	971	105	368	23	430	36	369	36
bg_{BTB}	457	43	478	44	199	18	220	19	226	17
cs_{PDT}	2128	213	2667	300	801	54	1015	84	958	115
en_{EWT}	867	52	902	59	263	11	297	16	293	11
et_{EDT}	391	37	391	37	118	12	119	12	114	13
fi_{TDT}	1019	90	1259	122	415	21	507	32	532	60
$fr_{\text{SEQ.}}$	486	78	497	81	168	19	188	20	163	20
it_{ISDT}	859	18	881	19	254	5	280	5	242	9
$lt_{\text{ALK.}}$	769	109	923	142	352	39	411	50	388	58
lv_{LVTB}	994	124	1236	176	396	43	486	60	515	55
$nl_{\text{ALP.}}$	758	34	845	39	291	8	311	10	306	3
pl_{PDB}	1130	112	1381	160	458	35	549	49	515	48
$ru_{\text{SYN.}}$	1212	108	1358	131	350	23	383	26	345	23
sk_{SNK}	452	74	490	85	220	32	233	40	199	32
$sv_{\text{TAL.}}$	560	110	563	115	203	25	225	32	204	27
ta_{TTB}	92	26	92	26	33	12	33	12	28	8
uk_{IU}	787	102	849	118	337	30	374	33	323	28
μ	858	84	982	105	376	27	465	38	379	35

Table 4: Number of generated labels in the training set and number of unseen labels in the development and evaluation sets. Average in the last row (μ).

	dev						id						ood					
	UF	LF	UM	LM	A	W	UF	LF	UM	LM	A	W	UF	LF	UM	LM	A	W
B$_2$	100	100	100	100	-	-	100	100	100	100	-	-	*100	*100	99.84	99.84	-	-
B$_3$	100	100	100	100	-	-	100	100	100	100	-	-	100	100	100	100	-	-
B6$_3$	99.38	99.38	79.85	79.85	-	-	99.51	99.51	83.62	83.62	-	-	99.66	99.66	89.89	89.89	-	-
B6$_4$	99.95	99.95	97.93	97.93	-	-	99.96	99.96	98.51	98.51	-	-	99.97	99.97	98.97	98.97	-	-
HB	100	100	100	100	-	-	100	100	100	100	-	-	100	100	100	100	-	-
B$_2$	95.56	95.10	59.04	56.26	94.53	85.40	95.30	94.57	56.52	52.27	**94.55**	82.12	92.58	92.15	52.79	49.54	91.43	75.93
B$_3$	**95.68**	**95.25**	60.11	56.80	**94.73**	85.51	95.28	94.67	56.45	52.48	94.53	84.29	**92.75**	92.19	53.76	**50.84**	**91.59**	79.82
B6$_3$	94.66	94.25	47.87	46.16	93.42	79.38	94.59	94.04	46.88	43.69	93.32	78.90	91.84	91.58	48.30	45.97	90.08	71.44
B6$_4$	95.62	95.13	59.16	56.03	93.66	82.32	**95.37**	**94.74**	55.39	51.77	93.44	76.86	92.67	**92.28**	53.76	50.41	90.45	73.25
HB	95.60	95.10	**61.35**	**57.45**	93.99	**90.01**	95.07	94.52	**57.16**	**53.05**	93.66	**89.66**	92.39	92.09	**54.14**	50.73	90.55	**85.55**
Biaf	95.93	95.76	54.79	51.83	-	-	95.51	95.45	50.00	46.95	-	-	92.82	92.92	48.08	45.65	-	-

Table 5: Performance on the English-DM (en_{DM}) dataset. The first and second row groups correspond to coverage and parsing metrics, respectively. An asterisk (*100) denotes coverage metrics that are near full coverage. Acronyms follow the notation used in Table 1.

	dev						id						ood					
	UF	LF	UM	LM	A	W	UF	LF	UM	LM	A	W	UF	LF	UM	LM	A	W
B$_2$	*100	*100	99.94	99.94	-	-	*100	*100	99.93	99.93	-	-	*100	*100	99.78	99.78	-	-
B$_3$	100	100	100	100	-	-	100	100	100	100	-	-	100	100	100	100	-	-
B6$_3$	97.73	97.73	46.45	46.45	-	-	97.56	97.56	42.55	42.55	-	-	98.24	98.24	63.33	63.33	-	-
B6$_4$	99.39	99.39	80.08	80.08	-	-	99.37	99.37	78.30	78.30	-	-	99.53	99.53	87.29	87.29	-	-
HB	100	100	100	100	-	-	100	100	100	100	-	-	100	100	100	100	-	-
B$_2$	96.38	95.40	55.26	46.75	93.54	85.37	96.08	95.27	54.33	47.02	93.52	83.00	94.76	93.80	56.90	51.22	91.96	81.11
B$_3$	**96.50**	**95.63**	56.62	**48.17**	**93.63**	89.96	**96.32**	95.50	57.30	49.65	**93.76**	89.27	**95.11**	**94.13**	58.79	53.49	**92.16**	82.58
B6$_3$	94.25	93.80	30.50	26.83	92.52	86.03	93.90	93.56	27.59	24.47	92.27	84.34	93.41	92.54	42.02	38.78	90.93	79.07
B6$_4$	95.95	95.13	48.70	41.25	92.34	90.15	95.66	95.01	48.01	42.27	91.81	87.70	94.45	93.59	53.70	49.70	90.22	81.29
HB	96.32	95.32	**57.68**	47.64	92.82	**96.61**	96.26	**95.59**	**59.08**	**51.84**	93.31	**95.48**	94.96	94.01	**60.09**	**54.52**	91.54	**92.53**
Biaf	96.69	95.87	55.73	47.10	-	-	96.64	96.13	55.96	48.51	-	-	95.75	94.97	60.03	54.73	-	-

Table 6: Performance in the English-PAS (en_{PAS}) dataset. Same notation as in Table 5.

	dev						id						ood					
	UF	LF	UM	LM	A	W	UF	LF	UM	LM	A	W	UF	LF	UM	LM	A	W
B$_2$	99.97	99.97	98.88	98.88	-	-	99.96	99.96	98.58	98.58	-	-	99.97	99.97	99.03	99.03	-	-
B$_3$	*100	*100	99.82	99.82	-	-	*100	*100	99.86	99.86	-	-	*100	*100	99.89	99.89	-	-
B6$_3$	99.97	99.97	98.94	98.94	-	-	99.98	99.98	98.94	98.94	-	-	99.96	99.96	98.81	98.81	-	-
B6$_4$	99.99	99.99	99.65	99.65	-	-	*100	*100	99.93	99.93	-	-	99.99	99.99	99.68	99.68	-	-
HB	100	100	100	100	-	-	100	100	100	100	-	-	100	100	100	100	-	-
B$_2$	93.74	86.93	48.76	18.50	92.44	75.64	92.53	85.33	45.60	15.82	91.29	73.52	92.42	86.09	52.46	26.66	90.47	72.79
B$_3$	94.17	87.21	51.89	19.03	92.73	81.97	93.10	85.95	47.09	17.09	91.52	79.77	92.81	**86.60**	54.68	27.37	90.81	79.09
B6$_3$	94.15	87.14	51.18	18.74	93.45	84.10	93.06	85.68	46.81	15.67	92.52	77.96	92.83	86.52	53.81	26.66	91.91	79.89
B6$_4$	**94.37**	**87.54**	**53.19**	**20.09**	**93.61**	86.89	**93.37**	**86.20**	**48.44**	**17.66**	**92.80**	82.55	**93.08**	86.56	**55.11**	26.18	**92.01**	83.84
HB	93.64	86.85	52.01	19.15	92.87	**91.21**	92.52	85.61	47.73	16.67	92.22	**88.66**	91.91	85.90	53.43	**27.53**	91.33	**89.02**
Biaf	94.64	87.91	47.75	19.56	-	-	93.70	86.84	44.47	16.60	-	-	93.12	87.00	50.84	26.55	-	-

Table 7: Performance in the English-PSD (en_{PSD}) dataset. Same notation as in Table 5.

	dev						id						ood					
	UF	LF	UM	LM	A	W	UF	LF	UM	LM	A	W	UF	LF	UM	LM	A	W
B$_2$	99.96	99.96	98.66	98.66	-	-	99.96	99.96	98.02	98.02	-	-	99.94	99.94	97.82	97.82	-	-
B$_3$	*100	*100	99.90	99.90	-	-	100	100	100	100	-	-	*100	*100	99.83	99.83	-	-
B6$_3$	99.98	99.98	99.35	99.35	-	-	99.98	99.98	99.22	99.22	-	-	99.97	99.97	98.91	98.91	-	-
B6$_4$	99.99	99.99	99.85	99.85	-	-	*100	*100	99.82	99.82	-	-	99.99	99.99	99.69	99.69	-	-
HB	100	100	100	100	-	-	100	100	100	100	-	-	100	100	100	100	-	-
B$_2$	94.49	90.84	48.16	31.69	91.87	76.10	93.75	90.02	45.81	28.80	91.34	72.98	91.85	83.12	49.00	19.29	87.95	71.32
B$_3$	94.44	90.73	47.91	31.99	91.85	76.87	93.60	90.03	45.99	29.40	91.08	75.31	91.82	83.07	48.99	19.23	87.73	71.23
B6$_3$	94.89	**91.29**	49.85	32.14	**93.05**	76.65	94.01	90.25	46.53	29.52	92.15	76.57	91.67	83.00	49.06	18.91	89.05	73.99
B6$_4$	**94.94**	91.15	**51.14**	**32.99**	93.00	82.45	**94.15**	**90.48**	**48.32**	**31.56**	**92.31**	80.56	**92.28**	**83.54**	**51.19**	**19.80**	**89.23**	78.53
HB	93.84	90.20	49.00	32.39	91.95	**86.97**	93.29	89.64	47.90	30.84	91.36	**87.28**	91.01	82.35	49.16	19.31	88.90	**85.60**
Biaf	95.30	92.07	44.03	30.20	-	-	94.61	91.21	43.11	27.90	-	-	92.55	83.81	44.18	17.76	-	-

Table 8: Performance in the Czech-PSD (cs_{PSD}) dataset. Same notation as in Table 5.

	dev						id					
	UF	LF	UM	LM	A	W	UF	LF	UM	LM	A	W
B$_2$	*100	*100	99.75	99.75	-	-	*100	*100	99.77	99.77	-	-
B$_3$	100	100	100	100	-	-	100	100	100	100	-	-
B6$_3$	96.65	96.65	43.69	43.69	-	-	96.57	96.57	46.79	46.79	-	-
B6$_4$	98.33	98.33	69.51	69.51	-	-	98.30	98.30	71.40	71.40	-	-
HB	100	100	100	100	-	-	100	100	100	100	-	-
B$_2$	**89.55**	87.35	29.80	26.89	86.00	59.08	89.25	87.20	34.78	31.72	85.93	62.18
B$_3$	89.53	**88.60**	30.94	28.16	**86.17**	64.49	**89.43**	**88.67**	35.77	32.70	**86.08**	66.03
B6$_3$	86.47	84.34	19.14	17.50	82.24	53.78	86.63	84.67	23.02	21.41	82.05	56.14
B6$_4$	88.34	87.92	25.29	22.91	81.28	53.56	88.08	87.72	29.52	27.18	81.08	57.47
HB	88.98	88.06	**32.09**	**29.14**	83.27	**67.49**	88.82	88.09	**37.63**	**34.35**	83.70	**74.77**
Biaf	91.38	90.67	31.43	28.48	-	-	91.02	90.38	37.14	34.06	-	-

Table 9: Performance in the Chinese-PAS (zh_{PAS}) dataset. Same notation as in Table 5.

	dev						id					
	UF	LF	UM	LM	A	W	UF	LF	UM	LM	A	W
B$_2$	99.98	99.98	98.46	98.46	-	-	99.97	99.97	98.09	98.09	-	-
B$_3$	*100	*100	99.78	99.78	-	-	*100	*100	99.85	99.85	-	-
B6$_3$	99.98	99.98	98.79	98.79	-	-	99.97	99.97	98.68	98.68	-	-
B6$_4$	99.99	99.99	99.45	99.45	-	-	99.98	99.98	99.56	99.56	-	-
HB	100	100	100	100	-	-	100	100	100	100	-	-
B$_2$	86.73	**82.62**	16.39	7.92	83.32	45.11	88.77	83.13	18.97	11.47	84.91	40.15
B$_3$	86.88	80.84	16.28	6.93	83.34	48.70	89.01	82.75	19.12	10.15	85.14	43.88
B6$_3$	87.37	81.38	18.26	8.36	84.15	58.07	88.89	82.72	20.59	11.62	**85.90**	51.46
B6$_4$	**87.70**	81.63	**19.47**	**8.58**	**84.45**	61.09	**89.20**	**83.17**	**22.79**	**13.24**	85.80	50.29
HB	87.37	81.29	18.70	8.36	82.87	**65.45**	88.90	82.54	21.32	11.47	84.03	**54.68**
Biaf	89.38	84.72	16.61	6.16	-	-	91.24	85.26	18.24	11.03	-	-

Table 10: Performance in the Arabic-PADT (ar_{PADT}) dataset. Same notation as in Table 5.

	dev						id					
	UF	LF	UM	LM	A	W	UF	LF	UM	LM	A	W
B$_2$	*100	*100	99.91	99.91	-	-	100	100	100	100	-	-
B$_3$	100	100	100	100	-	-	100	100	100	100	-	-
B6$_3$	99.98	99.98	99.10	99.10	-	-	99.99	99.99	99.64	99.64	-	-
B6$_4$	*100	*100	99.91	99.91	-	-	99.99	99.99	99.73	99.73	-	-
HB	100	100	100	100	-	-	100	100	100	100	-	-
B$_2$	94.72	93.05	65.20	46.10	93.04	82.21	95.74	93.97	64.61	47.76	93.01	81.15
B$_3$	94.76	91.56	63.86	43.86	92.85	78.27	95.72	92.89	65.05	47.94	93.04	79.65
B6$_3$	94.86	**93.11**	66.73	46.64	**93.69**	87.67	95.95	**94.29**	**68.82**	**51.52**	**94.31**	**88.77**
B6$_4$	**94.95**	91.83	**67.26**	46.55	93.66	87.46	**96.09**	93.32	68.28	50.45	94.18	86.83
HB	94.47	91.35	66.64	**47.09**	92.29	**91.38**	95.58	92.83	67.29	50.18	92.03	88.62
Biaf	95.63	93.84	65.92	47.35	-	-	96.69	94.92	66.94	48.75	-	-

Table 11: Performance in the Bulgarian-BTB (bg_{BTB}) dataset. Same notation as in Table 5.

	dev						id					
	UF	LF	UM	LM	A	W	UF	LF	UM	LM	A	W
B$_2$	99.94	99.94	98.31	98.31	-	-	99.93	99.93	97.94	97.94	-	-
B$_3$	99.99	99.99	99.63	99.63	-	-	*100	*100	99.87	99.87	-	-
B6$_3$	99.93	99.93	98.53	98.53	-	-	99.93	99.93	98.39	98.39	-	-
B6$_4$	99.98	99.98	99.34	99.34	-	-	99.97	99.97	99.16	99.16	-	-
HB	100	100	100	100	-	-	100	100	100	100	-	-
B$_2$	92.65	89.57	53.37	41.64	88.12	61.33	93.01	90.35	53.38	42.57	88.80	61.50
B$_3$	92.77	89.94	54.25	42.89	88.22	63.63	93.28	90.50	54.41	43.47	89.20	62.46
B6$_3$	93.16	90.28	57.77	44.94	89.91	71.08	**93.72**	**91.03**	**58.71**	**46.24**	**91.13**	76.32
B6$_4$	**93.26**	**90.40**	**58.28**	**46.19**	**89.93**	73.44	93.64	90.81	58.46	45.47	90.91	78.91
HB	92.41	89.62	56.89	44.79	88.69	**83.84**	92.71	90.10	56.91	45.59	89.26	**80.48**
Biaf	94.48	91.63	58.21	45.16	-	-	95.09	92.33	58.26	45.79	-	-

Table 12: Performance in the Finnish-TDT (fi_{TDT}) dataset. Same notation as in Table 5.

	dev						id					
	UF	LF	UM	LM	A	W	UF	LF	UM	LM	A	W
B$_2$	*100	*100	99.76	99.76	-	-	*100	*100	99.56	99.56	-	-
B$_3$	100	100	100	100	-	-	100	100	100	100	-	-
B6$_3$	99.97	99.97	98.30	98.30	-	-	99.97	99.97	98.46	98.46	-	-
B6$_4$	99.99	99.99	99.27	99.27	-	-	100	100	100	100	-	-
HB	100	100	100	100	-	-	100	100	100	100	-	-
B$_2$	93.54	92.27	40.53	35.44	90.94	54.45	93.72	92.98	41.01	37.72	90.09	52.60
B$_3$	94.01	92.15	39.81	33.01	91.04	49.89	93.87	92.47	41.23	38.16	90.67	50.14
B6$_3$	**95.49**	**94.04**	**50.97**	**42.48**	**93.10**	64.42	94.90	**93.98**	50.00	45.18	92.26	63.37
B6$_4$	95.23	93.37	49.03	41.26	93.06	70.25	**95.22**	93.88	**54.17**	**47.81**	**92.43**	70.09
HB	94.49	92.49	48.54	40.29	91.53	**72.61**	94.50	92.97	51.10	44.96	89.93	**73.62**
Biaf	96.47	94.44	51.21	40.78	-	-	96.22	94.91	52.41	47.59	-	-

Table 13: Performance in the French-SEQUOIA ($fr_{\text{SEQ.}}$) dataset. Same notation as in Table 5.

	dev						id					
	UF	LF	UM	LM	A	W	UF	LF	UM	LM	A	W
B$_2$	100	100	100	100	-	-	*100	*100	99.79	99.79	-	-
B$_3$	100	100	100	100	-	-	100	100	100	100	-	-
B6$_3$	100	100	100	100	-	-	*100	*100	99.59	99.59	-	-
B6$_4$	100	100	100	100	-	-	100	100	100	100	-	-
HB	100	100	100	100	-	-	100	100	100	100	-	-
B$_2$	94.40	92.60	53.37	43.79	92.43	75.31	94.97	93.16	54.98	45.44	92.71	67.81
B$_3$	94.36	92.40	55.14	44.50	92.57	77.09	95.53	93.47	56.22	45.02	93.21	72.97
B6$_3$	94.62	92.74	57.27	46.63	93.06	78.68	95.46	93.51	59.96	47.93	93.93	78.00
B6$_4$	**94.95**	**93.01**	**59.40**	**49.11**	**93.10**	**85.41**	**95.58**	93.52	**60.37**	48.76	**93.95**	81.19
HB	94.50	92.69	59.40	48.40	91.93	84.78	95.55	**93.75**	59.96	**49.38**	91.78	**84.99**
Biaf	95.40	93.35	54.26	44.33	-	-	96.31	94.36	57.47	48.13	-	-

Table 14: Performance in the Italian-ISDT (it_{ISDT}) dataset. Same notation as in Table 5.

	dev						id					
	UF	LF	UM	LM	A	W	UF	LF	UM	LM	A	W
B$_2$	99.98	99.98	98.54	98.54	-	-	99.92	99.92	97.22	97.22	-	-
B$_3$	100	100	100	100	-	-	99.99	99.99	99.56	99.56	-	-
B6$_3$	99.98	99.98	98.87	98.87	-	-	99.97	99.97	98.68	98.68	-	-
B6$_4$	100	100	100	100	-	-	99.99	99.99	99.71	99.71	-	-
HB	100	100	100	100	-	-	100	100	100	100	-	-
B$_2$	86.72	84.99	26.58	18.48	82.70	39.08	85.56	83.50	29.97	18.57	79.13	41.71
B$_3$	88.04	83.91	28.04	20.42	82.66	35.94	86.23	80.79	31.58	19.74	78.78	36.63
B6$_3$	88.74	**86.54**	32.09	21.23	85.07	57.52	87.05	**84.85**	36.99	21.93	**81.27**	59.38
B6$_4$	**89.24**	85.18	**34.04**	**22.20**	**85.40**	**63.11**	**87.59**	81.75	**37.57**	**22.37**	80.89	61.42
HB	86.72	82.35	30.31	21.07	83.04	63.11	84.54	79.12	31.87	20.32	78.04	**62.30**
Biaf	91.65	87.43	30.96	21.56	-	-	89.63	83.82	33.33	20.47	-	-

Table 15: Performance in the Lithuanian-ALKSNIS ($lt_{\text{ALK.}}$) dataset. Same notation as in Table 5.

	dev						id					
	UF	LF	UM	LM	A	W	UF	LF	UM	LM	A	W
B$_2$	99.95	99.95	97.96	97.96	-	-	99.95	99.95	98.41	98.41	-	-
B$_3$	*100	*100	99.82	99.82	-	-	99.99	99.99	99.78	99.78	-	-
B6$_3$	99.92	99.92	97.54	97.54	-	-	99.94	99.94	98.03	98.03	-	-
B6$_4$	99.97	99.97	99.22	99.22	-	-	99.98	99.98	99.23	99.23	-	-
HB	100	100	100	100	-	-	100	100	100	100	-	-
B$_2$	91.90	88.83	48.20	37.32	86.56	60.15	90.83	87.80	47.94	37.47	86.65	65.00
B$_3$	92.12	89.01	48.74	37.56	86.42	57.79	90.47	87.12	46.19	35.27	86.01	60.67
B6$_3$	92.09	89.13	50.60	38.40	**88.33**	71.54	90.93	87.78	49.20	38.29	**88.29**	75.01
B6$_4$	**92.68**	**89.58**	**52.70**	**39.96**	88.09	71.39	**91.51**	**88.30**	**51.89**	**39.99**	88.21	73.76
HB	91.58	88.73	50.54	39.30	85.97	**77.99**	90.54	87.46	49.75	38.51	86.25	**82.57**
Biaf.	93.71	90.73	50.36	38.76	-	-	93.18	89.84	51.84	39.66	-	-

Table 16: Performance in the Latvian-LVTB (lv_{LVTB}) dataset. Same notation as in Table 5.

	dev						id					
	UF	LF	UM	LM	A	W	UF	LF	UM	LM	A	W
B$_2$	99.99	99.99	99.54	99.54	-	-	99.99	99.99	99.65	99.65	-	-
B$_3$	*100	*100	99.98	99.98	-	-	100	100	100	100	-	-
B6$_3$	*100	*100	99.86	99.86	-	-	*100	*100	99.85	99.85	-	-
B6$_4$	*100	*100	99.94	99.94	-	-	*100	*100	99.95	99.95	-	-
HB	100	100	100	100	-	-	100	100	100	100	-	-
B$_2$	94.72	92.95	59.61	50.15	92.41	82.34	95.23	93.74	61.76	53.12	93.24	81.90
B$_3$	94.84	93.11	60.10	50.88	92.38	80.83	95.15	93.64	61.52	52.47	93.22	82.18
B6$_3$	94.81	93.06	61.47	51.50	93.10	85.44	95.30	93.86	63.21	54.01	**93.99**	87.52
B6$_4$	**94.98**	**93.19**	62.07	**52.05**	**93.15**	88.80	**95.39**	**93.93**	63.20	54.00	93.99	89.74
HB	94.76	92.96	**62.20**	51.91	91.67	**90.45**	95.01	93.54	**63.32**	**54.24**	92.46	**91.81**
Biaf	95.59	93.81	60.09	50.65	-	-	95.96	94.45	60.68	51.93	-	-

Table 17: Performance in the Russian-SYNTAGRUS ($ru_{\text{SYN.}}$) dataset. Same notation as in Table 5.

	dev						id					
	UF	LF	UM	LM	A	W	UF	LF	UM	LM	A	W
B$_2$	99.99	99.99	99.53	99.53	-	-	99.99	99.99	99.72	99.72	-	-
B$_3$	100	100	100	100	-	-	100	100	100	100	-	-
B6$_3$	*100	*100	99.91	99.91	-	-	99.99	99.99	99.62	99.62	-	-
B6$_4$	100	100	100	100	-	-	*100	*100	99.91	99.91	-	-
HB	100	100	100	100	-	-	100	100	100	100	-	-
B$_2$	95.13	92.10	64.25	51.32	91.67	74.42	94.92	92.08	63.71	50.90	90.96	72.80
B$_3$	95.39	92.20	63.96	50.00	91.91	73.41	94.69	91.83	61.64	49.29	90.90	74.11
B6$_3$	94.92	92.06	66.70	52.08	92.90	79.81	**95.43**	**92.51**	66.92	52.87	92.63	80.16
B6$_4$	**95.68**	**92.81**	**69.25**	53.40	**93.24**	84.79	95.30	92.41	**67.48**	53.16	**92.66**	83.75
HB	94.87	92.31	66.89	**54.15**	91.95	**88.42**	94.62	92.11	66.35	**54.85**	91.25	**85.19**
Biaf	96.78	93.97	68.77	54.43	-	-	96.81	93.99	68.61	54.38	-	-

Table 18: Performance in the Slovak-SNK (sk_{SNK}) dataset. Same notation as in Table 5.

	dev						id					
	UF	LF	UM	LM	A	W	UF	LF	UM	LM	A	W
B$_2$	*100	*100	99.80	99.80	-	-	*100	*100	99.84	99.84	-	-
B$_3$	100	100	100	100	-	-	100	100	100	100	-	-
B6$_3$	99.97	99.97	98.21	98.21	-	-	99.98	99.98	99.34	99.34	-	-
B6$_4$	99.99	99.99	99.40	99.40	-	-	*100	*100	99.92	99.92	-	-
HB	100	100	100	100	-	-	100	100	100	100	-	-
B$_2$	89.60	87.01	34.13	28.57	85.49	53.70	92.42	89.90	45.69	38.64	88.05	60.85
B$_3$	89.26	86.41	34.33	26.98	85.39	52.42	92.01	89.50	44.63	38.23	88.33	59.94
B6$_3$	**91.21**	**88.54**	40.67	32.94	**87.76**	57.28	**93.32**	**91.12**	**53.49**	**45.28**	**90.77**	69.53
B6$_4$	90.93	87.92	**41.07**	**33.13**	87.41	56.62	93.14	90.97	50.78	43.23	90.53	68.07
HB	89.84	87.06	40.08	31.75	84.73	**69.08**	92.16	89.87	49.71	42.00	88.30	**76.20**
Biaf	93.04	89.97	37.10	28.77	-	-	94.59	92.05	48.89	41.43	-	-

Table 19: Performance in the Swedish-TALBANKEN ($sv_{\text{TAL.}}$) dataset. Same notation as in Table 5.

	dev						id					
	UF	LF	UM	LM	A	W	UF	LF	UM	LM	A	W
B_2	100	100	100	100	-	-	100	100	100	100	-	-
B_3	100	100	100	100	-	-	100	100	100	100	-	-
$B6_3$	100	100	100	100	-	-	100	100	100	100	-	-
$B6_4$	100	100	100	100	-	-	100	100	100	100	-	-
HB	100	100	100	100	-	-	100	100	100	100	-	-
B_2	78.22	67.10	11.25	**6.25**	72.09	27.43	74.15	61.89	7.50	1.67	74.77	44.15
B_3	78.22	67.10	11.25	6.25	72.09	27.43	74.15	61.89	7.50	1.67	74.77	44.15
$B6_3$	**79.69**	**69.31**	**15.00**	6.25	76.81	**63.29**	**78.18**	**66.08**	**11.67**	2.50	77.01	55.27
$B6_4$	79.69	69.31	15.00	6.25	76.81	63.29	78.18	66.08	11.67	2.50	77.01	55.27
HB	76.87	66.36	15.00	6.25	**77.18**	62.86	75.33	63.67	11.67	**3.33**	**77.49**	**74.77**
Biaf	79.43	67.80	8.75	5.00	-	-	76.69	64.83	9.17	2.50	-	-

Table 20: Performance in the Tamil-TTB (ta_{TTB}) dataset. Same notation as in Table 5.

	dev						id					
	UF	LF	UM	LM	A	W	UF	LF	UM	LM	A	W
B_2	99.99	99.99	99.40	99.40	-	-	99.99	99.99	99.10	99.10	-	-
B_3	100	100	100	100	-	-	100	100	100	100	-	-
$B6_3$	99.99	99.99	99.55	99.55	-	-	99.99	99.99	99.10	99.10	-	-
$B6_4$	*100	*100	99.85	99.85	-	-	*100	*100	99.78	99.78	-	-
HB	100	100	100	100	-	-	100	100	100	100	-	-
B_2	92.54	90.53	39.58	33.78	87.49	55.52	92.39	89.94	40.92	34.30	86.84	49.92
B_3	92.25	89.88	38.99	32.59	87.60	54.14	92.27	89.96	39.01	32.74	86.61	52.20
$B6_3$	**93.20**	**91.02**	43.90	36.01	**89.10**	66.24	92.85	90.38	44.62	37.33	88.44	63.98
$B6_4$	93.07	90.88	**45.68**	**36.61**	88.90	68.32	**93.03**	**90.59**	**46.30**	**37.89**	**88.58**	61.34
HB	91.97	89.84	42.86	35.86	86.50	**75.81**	91.81	89.50	43.72	37.44	85.37	**67.84**
Biaf	94.71	92.28	43.15	34.38	-	-	94.50	92.17	44.96	38.34	-	-

Table 21: Performance in the Ukrainian-IU (uk_{IU}) dataset. Same notation as in Table 5.