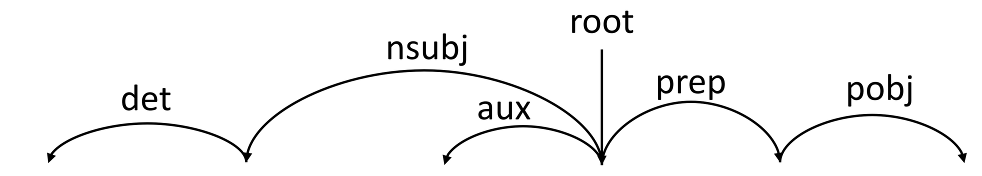
Hierarchical Bracketing Encodings for Dependency Parsing as Tagging

Ana Ezquerro, David Vilares, Anssi Yli-Jyrä, Carlos Gómez-Rodríguez

Dependency Parsing as Sequence Labeling

Contribution: Novel SL approach for projective dependency parsing (12 labels).



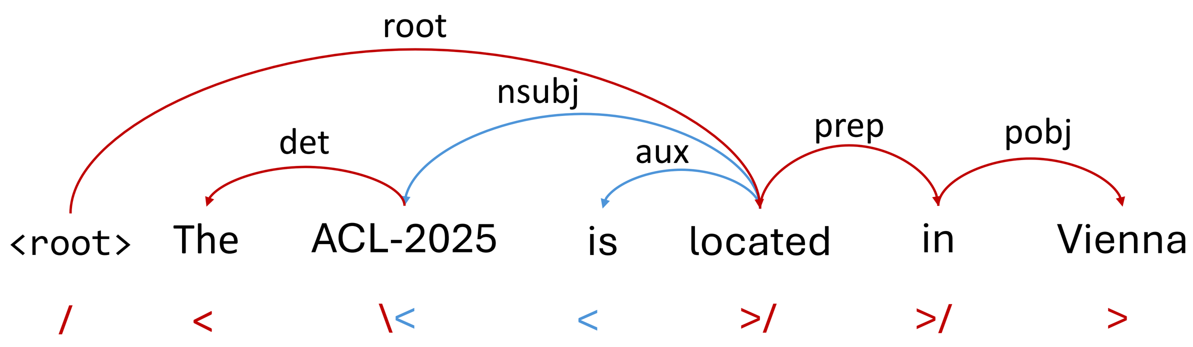
	The	ACL-2025	is	located	in	Vienna	**Label Space**
Bracketing: Strzyz et al. (2020)	<	\<	<	\\>/	>/	>	unbounded
Hexatagging: Amini et al. (2023)	↗↗R	↖↗R	↖↗R	↖↗L	↗↖L	↖	bounded (8)
4-bit: Gómez-Rodríguez et al. (2023)	0100	0110	0000	1111	1101	1100	bounded (16)
NEW! (ours):	<	\<	<	>/	>/	>	bounded (12)

Preliminaries

Let $G = (W, A)$ be a dependency tree. The *proper rope cover* is $R \subseteq A$ such that:

1. Every $a \in A - R$ leans on some $a' \in R$.
2. No $a \in R$ leans on any other $a' \in R - \{a\}$.

Yli-Jyrä (2019) proves that the proper rope cover of a graph is unique.



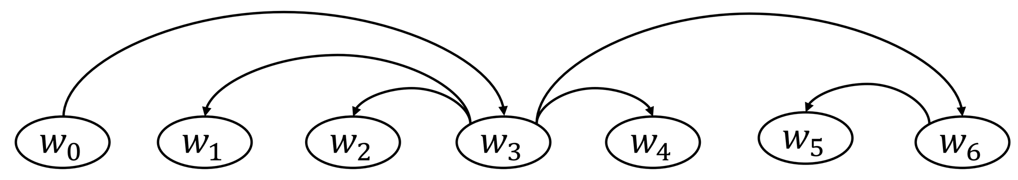
Superbrackets
> \ / <

Auxiliary brackets
> <

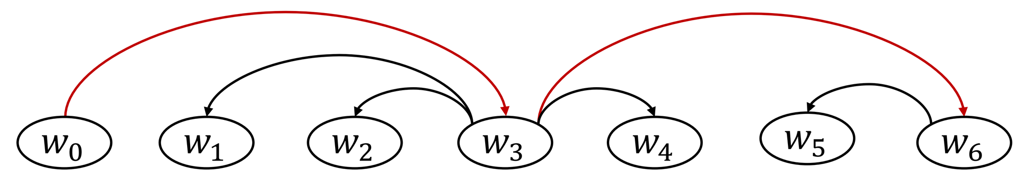
Optimal Hierarchical Bracketing Encoding

1. Identify proper rope cover (R).
2. Assign superbrackets (>, \, /, <).
3. For each arc $a \in R$, find those arcs that lean on a (auxiliary arcs) and assign auxiliary brackets (>, <).

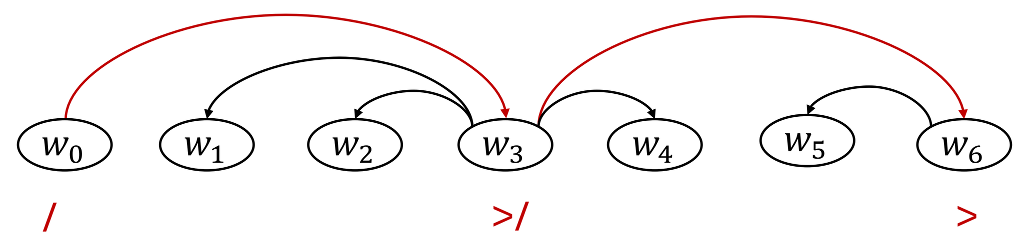
Projective dependency tree



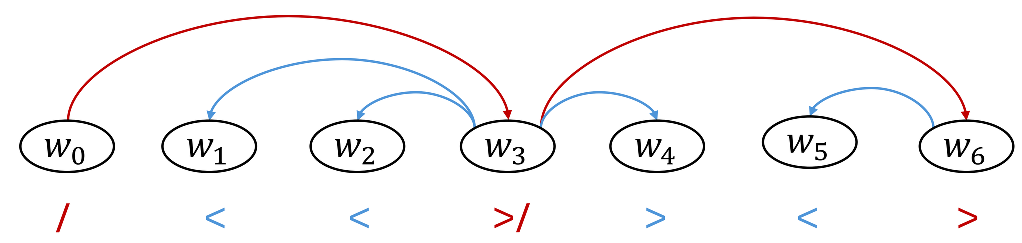
Step 1: Identify R



Step 2: Assign superbrackets



Step 3: Assign auxiliary brackets



Note that $(w_0 \rightarrow w_3) \in R$ is represented with / and >, but $(w_3 \rightarrow w_2)$ is represented with < and >.

The Bounded Space of Optimal Hierarchical Brackets (OHB)

Proof from 4-bit encoding (2^4 = 16 labels)

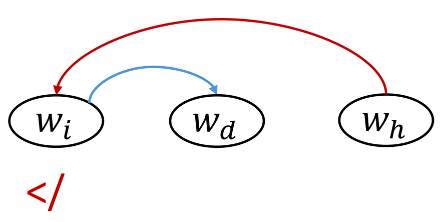
- $b_0 = w_i$ has a left head.
- $b_1 = w_i$ is the outermost dependent.
- $b_2 = w_i$ has left dependents.
- $b_3 = w_i$ has right dependents.

from Gómez-Rodríguez et al. (2023).

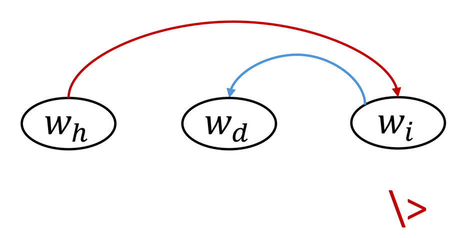
b_0b_1 \ b_2b_3	00	01	10	11
00	<	</	\<	\</
01	<	</	\<	\</
10	>	>/	\>	\>/
11	>	>/	\>	\>/

But {\</, \>/, </, \>} **cannot** occur in OHB since no $a \in R$ leans on $a' \in R - \{a\}$.

Never happens since $(w_i \rightarrow w_d)$ is auxiliary.



Never happens since $(w_d \leftarrow w_i)$ is auxiliary.

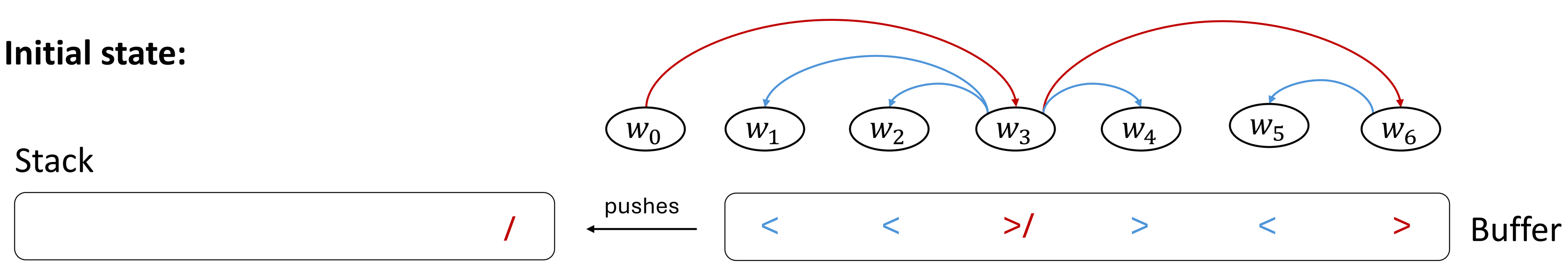


Note that \</ and \>/ cannot occur since they contain </ and \>.

Optimal Hierarchical Bracketing Decoding

- Similar to the standard bracketing decoding (Strzyz et al., 2020).
- Stack-based system that parses the bracket sequence:
  - Matching opening (<, /) with closing (\, >) super brackets.
  - Matching < with (>, \) and > with (<, /).
- The brackets are processed from left to right until the buffer is empty.
- In the initial state the first element in the stack is always /.

Initial state:



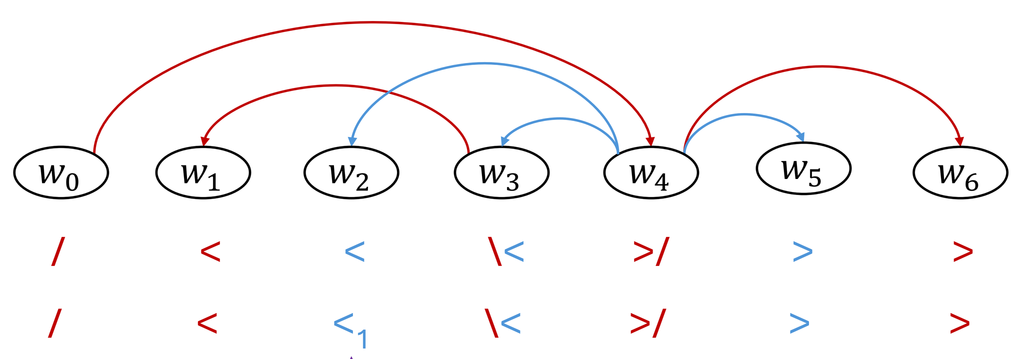
Example: < matches > to recover $(w_2 \leftarrow w_3)$

Non-Projective Extension for Hierarchical Bracketing

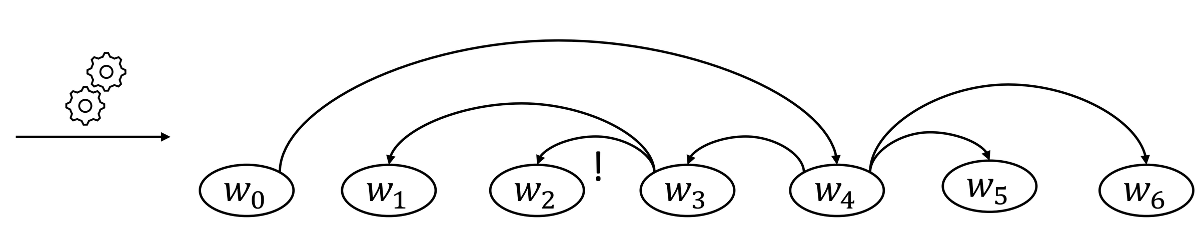
The OHB encoding does not recover crossing arcs.
Solution: Add indices to closing superbrackets (\, >) and auxiliary brackets (>, \, /, <) to skip matches during the decoding step.
Drawback: The label space becomes unbounded.

Use subindex 1 to indicate that $<_1$ matches with the second superbracket found (> at w_4).

Non-Projective Dependency Tree



Decoded Tree with the projective decoding



Error: $(w_2 \leftarrow w_4)$ is not recovered since it crosses $(w_1 \leftarrow w_3)$

Experiments and Results

- PTB and UD (9 languages) with XLM✧/XLNet ☐.
- Baselines: Hexatagging (H^+) and Biaffine (DM).
- 4-bit (B_4) and projective OHB (O_p) with pseudo-projectivity (+).
- 7-bit (B_7) and non-projective OHB (O_{np}).

Close performance:

- LAS: $O_p < O_{np} < B_4 < O_p^+ < B_4^+ < B_7$.
- LCM: $O_p^+ < B_4 < O_p < B_4^+ < O_{np} < B_7$.

Code and Materials: https://github.com/anaezquerro/separ.

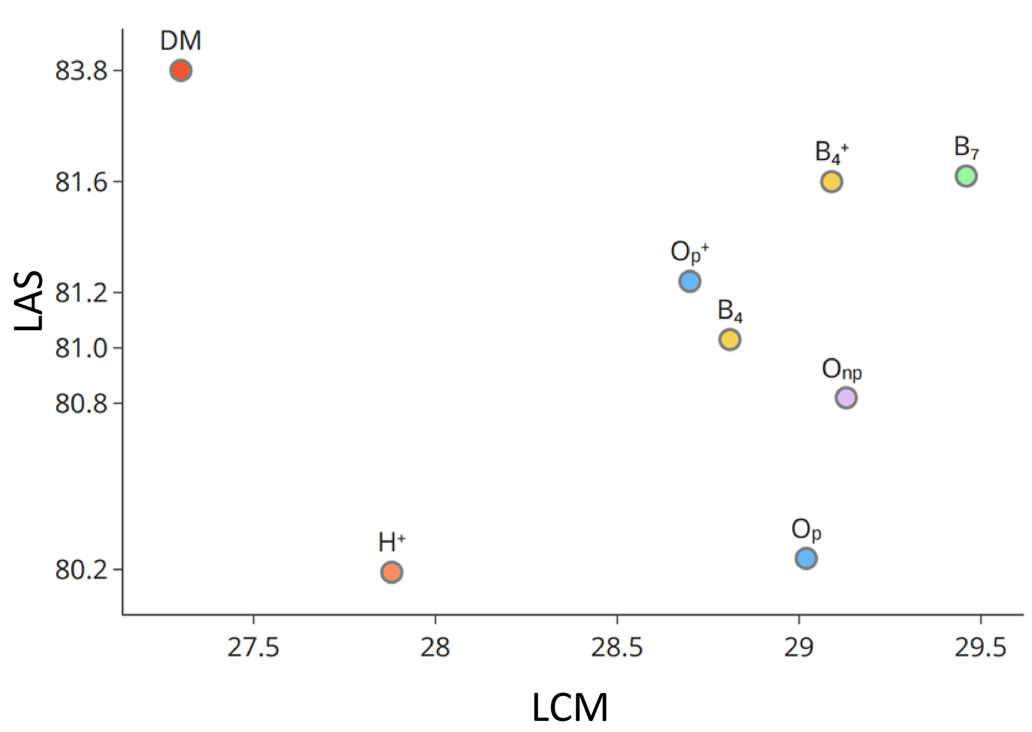
But O_p/O_{np} are faster (compact label space).

Label Space Analysis:

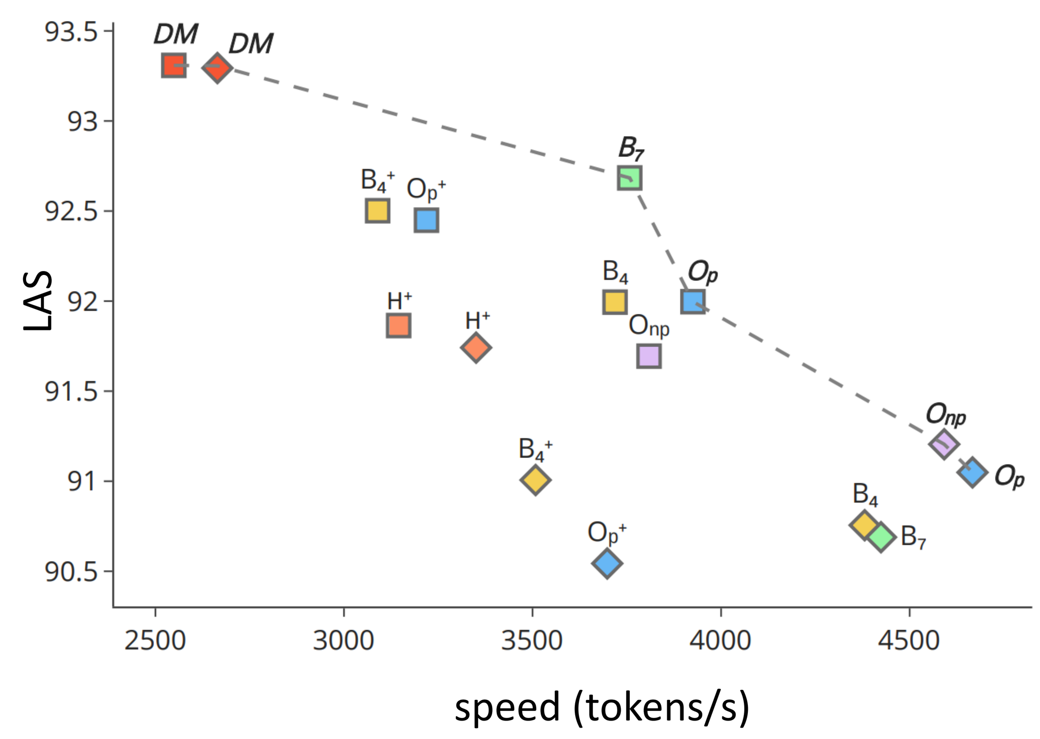
- B_4 always requires 16 labels, while O_p requires 12 labels.
- Except for Ancient-Greek, O_{np} requires less labels than B_7.

	#trees	#labels					#rels		#indices			
		B_4	O_P	B_7	O_{NP}	H^+	O_P	O_P^+	0	1	2	≥ 3
grc	13.9k	16	12	103	168	8	26	248	42.87	51.32	5.50	0.25
en	16.6	16	12	60	55	8	53	169	97.98	1.98	0.04	0
fi	15.1k	16	12	59	56	8	47	150	96.66	2.40	0.93	0
fr	16.3k	16	12	51	50	8	56	142	96.63	3.23	0.14	0
he	6.1k	16	12	37	30	8	37	52	99.22	0.75	0.03	0
ru	5k	16	12	60	52	8	42	153	95.27	4.31	0.42	0
ta	600	16	12	24	17	8	29	35	98.33	1.67	0	0
ug	3.5k	16	12	40	35	8	40	67	93.40	6.57	0.03	0
wo	2.1k	16	12	40	27	8	38	62	97.15	2.85	0	0
PTB	44k	16	12	22	21	8	45	46	99.90	0.10	0	0

Performance (average)



Pareto Front (English-EWT)



We acknowledge grants SCANNER-UDC (PID2020-113230RB-C21) funded by MICIU/AEI/10.13039/501100011033; GAP (PID2022-139308OA-I00) funded by MICIU/AEI/10.13039/501100011033/ and ERDF, EU; LATCHING (PID2023-147129OB-C21) funded by /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). We also extend our gratitude to CESGA, the supercomputing center of Galicia, for granting us access to its resources.Furthermore, we acknowledge the Faculty of Agriculture and Forestry of the University of Helsinki, as well as projects "Theory of Computational Logics" (352420) and "XAILOG" (345612, 345633) funded by the Research Council of Finland for the continued support of the third author during the multistage writing process.