Some Dependency Parsing Work

Mark Anderson

Overview

Developing

- Chunk-and-Pass
- Distillation

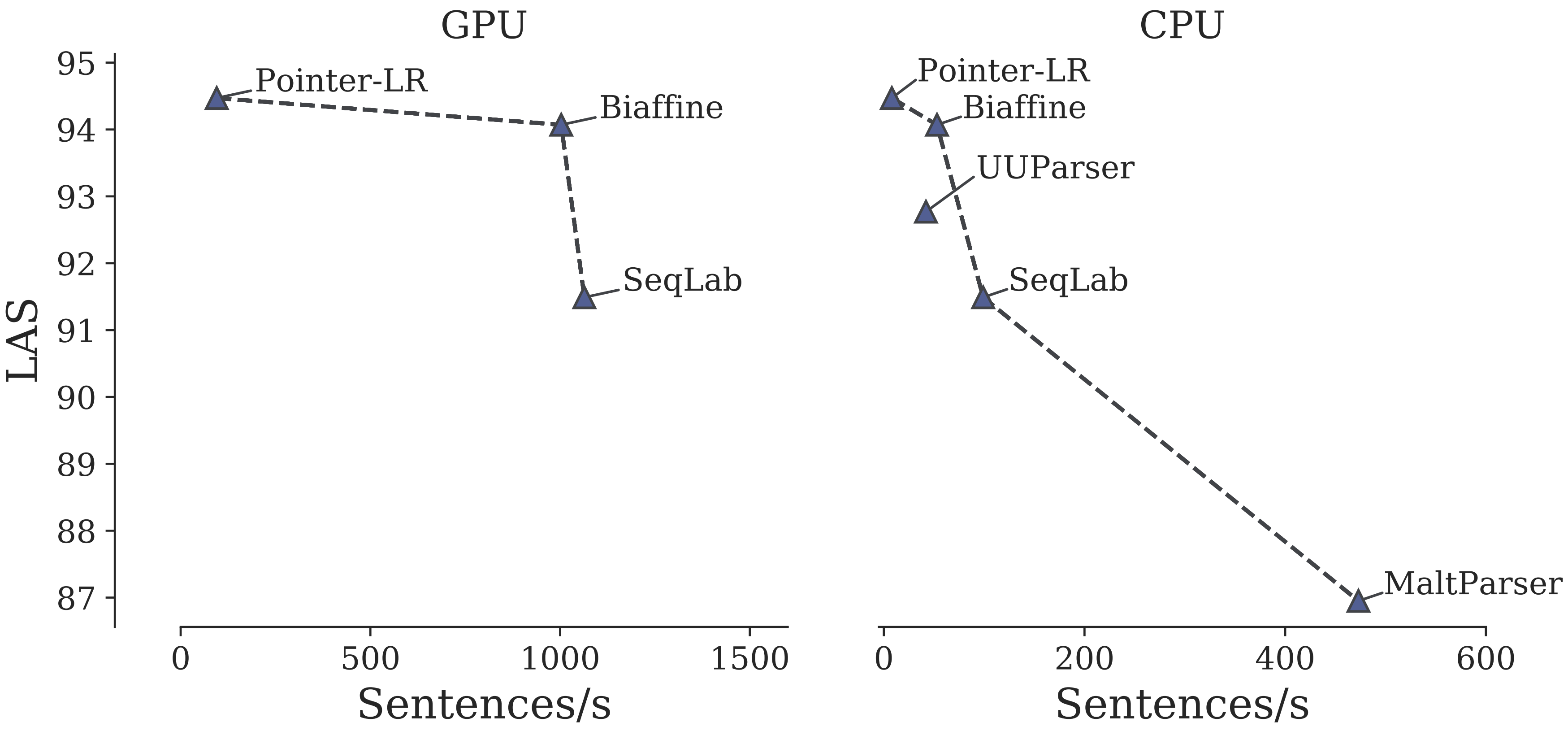
Evaluating

- Edge displacement
- POS tags

Part I

Developing Parsers

Pareto Front for PTB (WSJ)



Chunk and Pass

Chunk and Pass

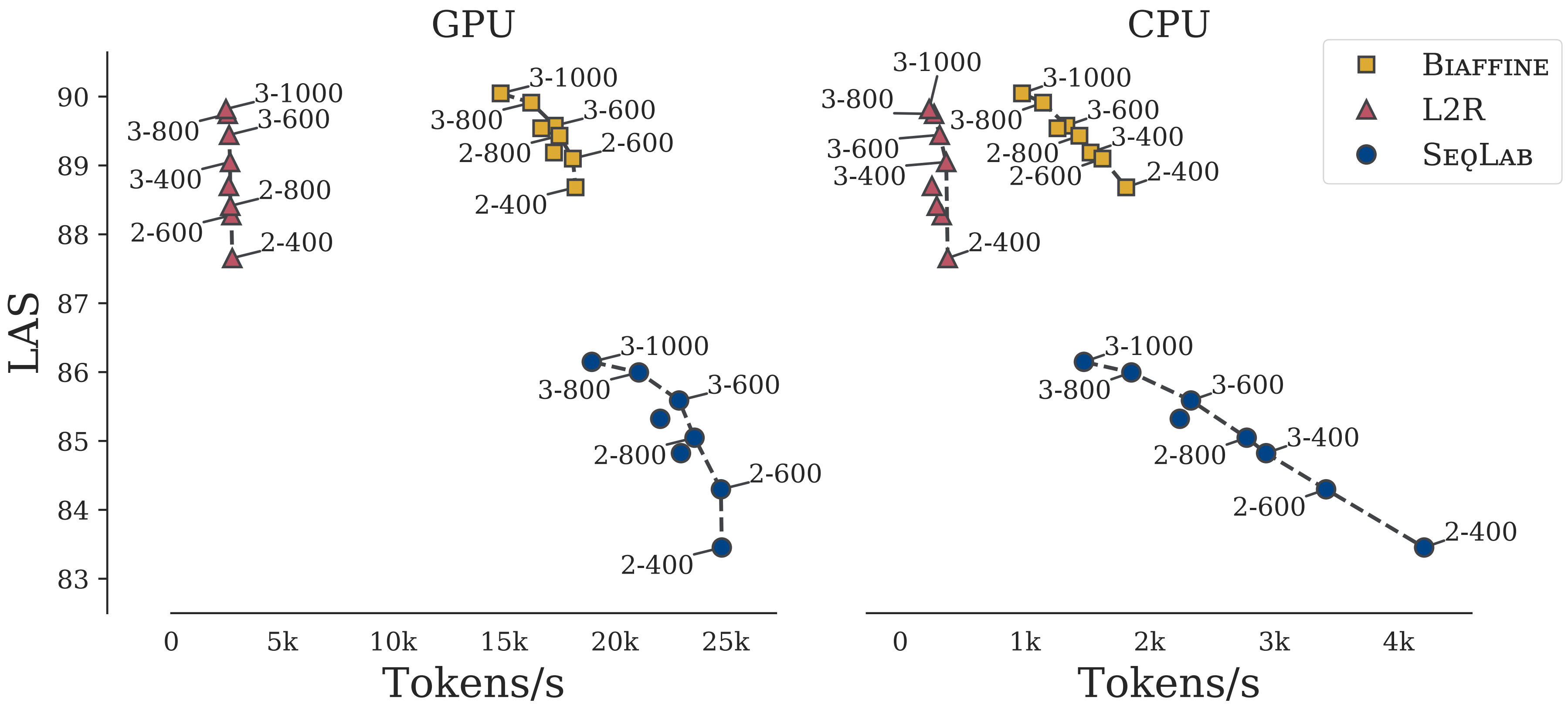
"Now-or-never" bottleneck[1]

3 steps

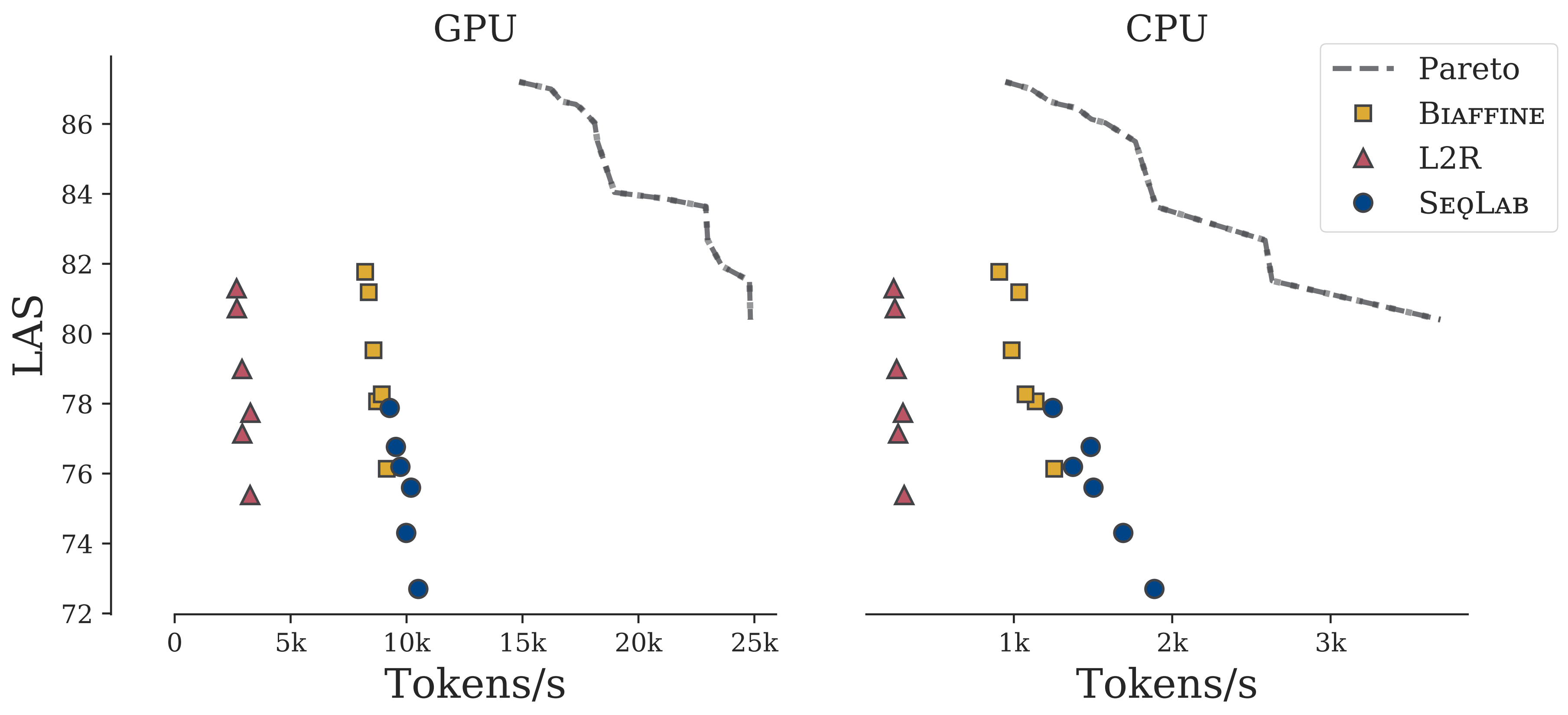
1. Shallow parse (chunk) --> Compressed data and shallow syntax
2. Parse chunks --> Higher-order syntactic relations
3. Collate --> Combined 1 and 2 for full parse

[1] Christiansen, M.H. and Chater N., *The Now-or-Never bottleneck: A fundamental constraint on language*, 2015

Pareto Front for UD (zh, hi, ko, pl)



Pareto Front for UD (zh, hi, ko, pl)



Chunking — Each word is labelled, **B**, **I**, or **O**.

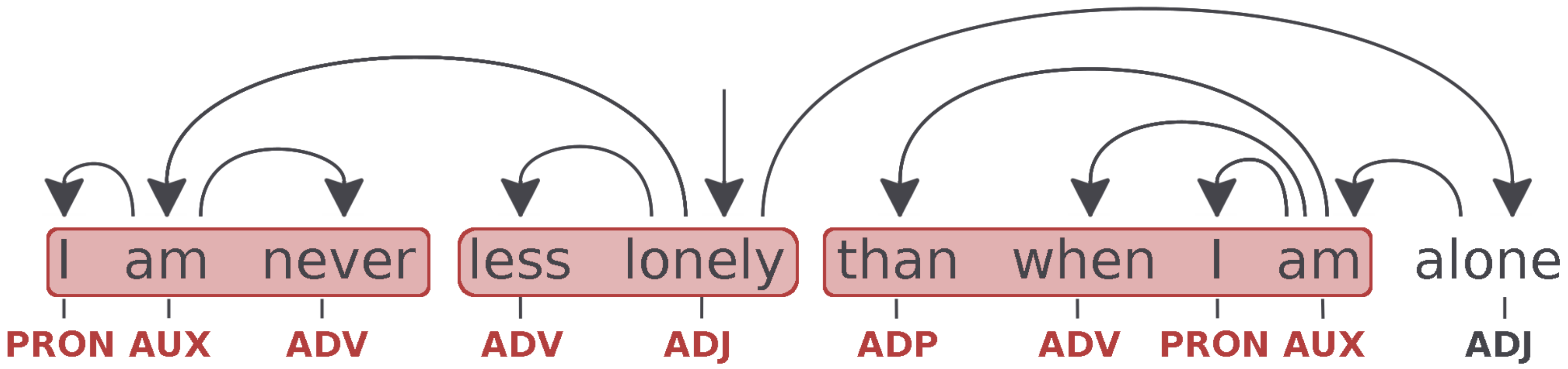
B	**I**	**O**
A token that begins a chunk.	A token inside of a chunk.	Anything outside of a chunk.
Suffixed with chunk phrase type.	Also suffixed with chunk phrase type.	
E.g. B-NP for a noun phrase.	E.g. I-VP for a verb phrase.	

Extracting chunks

1. Evolutionary technique (slow...)
2. Information theory

Chunk candidate criteria

1. The components are syntactically linked
2. There is only one level of dependency (one head and its dependents)
3. The components are continuous.
4. No dependents within a chunk has a dependent outwith the chunk.


I am never less lonely than when I am alone
PRON AUX ADV ADV ADJ ADP ADV PRON AUX ADJ

(DET ADJ NOUN)

(PRON AUX ADV)

(PART VERB)

(ADP ADV PRON AUX)

(SCONJ ADV VERB)

(AUX AUX VERB)

(PRON PROPN VERB)

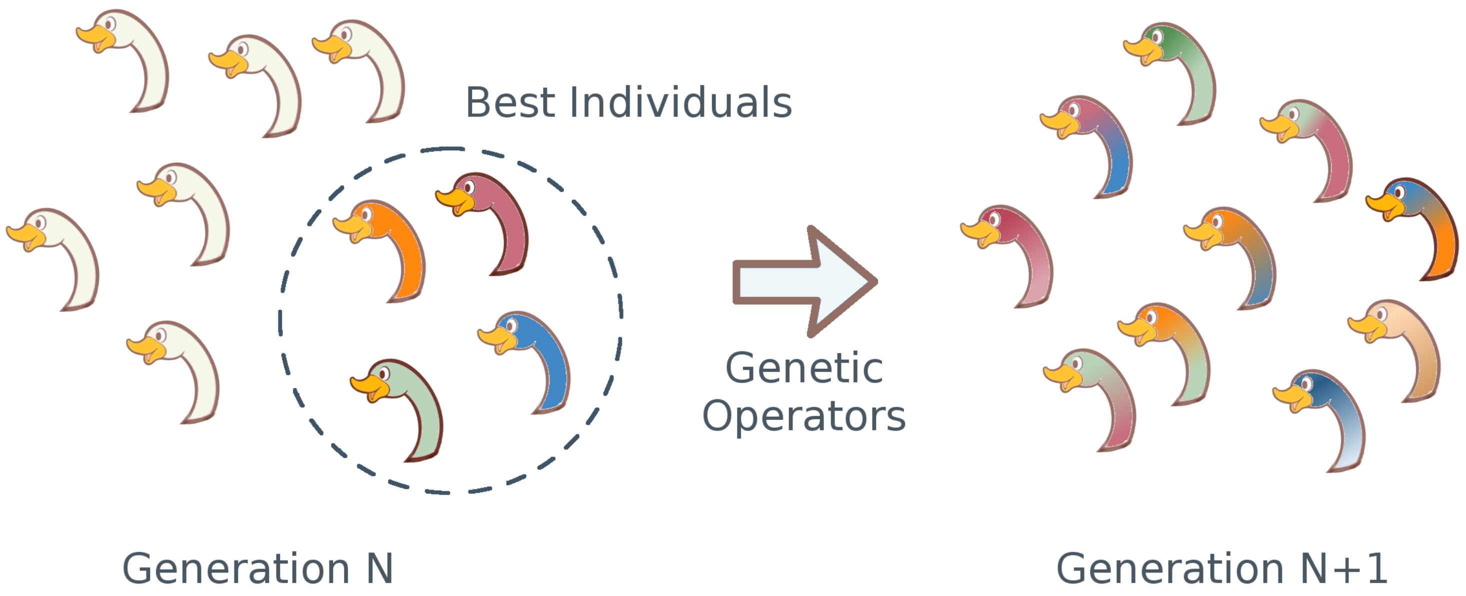
(CCONJ PRON AUX DET ADJ NOUN)

Extract 2615 unique rules from UD English EWT treebank v2.3

512 occur more than 5 times.

1.34×10^{154} different rule sets.

Evolutionary Search



Evolutionary Search

Individual =[0,1,0,0,1 .... 0,1]

(DET ADJ NOUN) **(PRON AUX ADV)** **(PART VERB)**

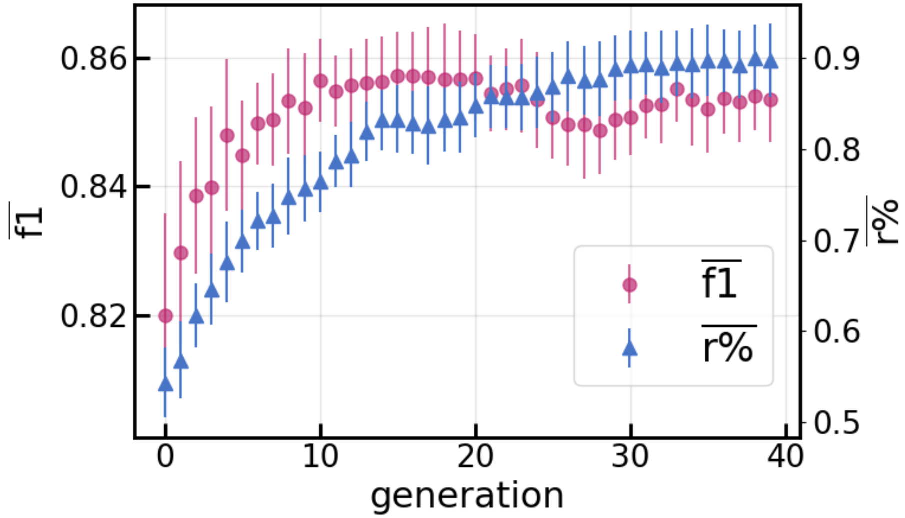
Fitness = Chunking F1-score
+ 0.5 x proportion of
max compression

Proportion of max compression

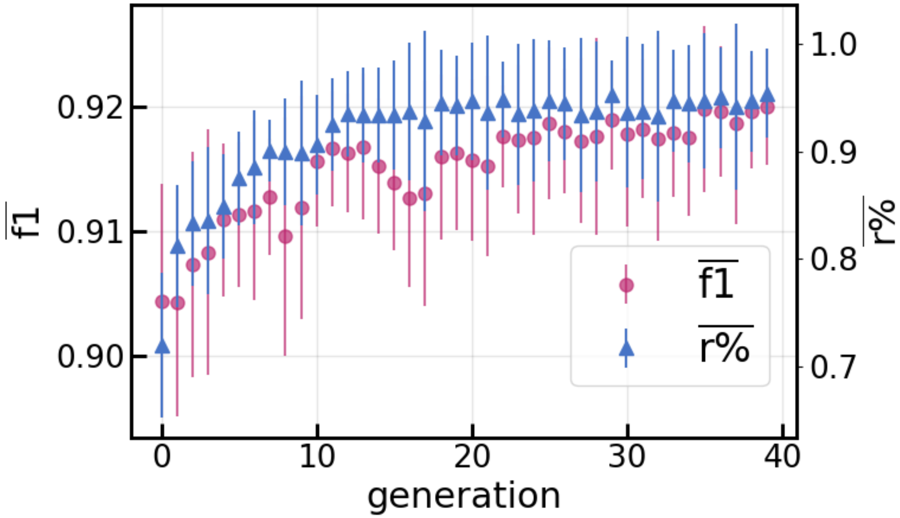
$$r = \frac{N_{tokens}}{N_{tokens} + N_{chunks}}$$

$$r_{prop} = \frac{r_{subset} - 1}{r_{all} - 1}$$

English-EWT



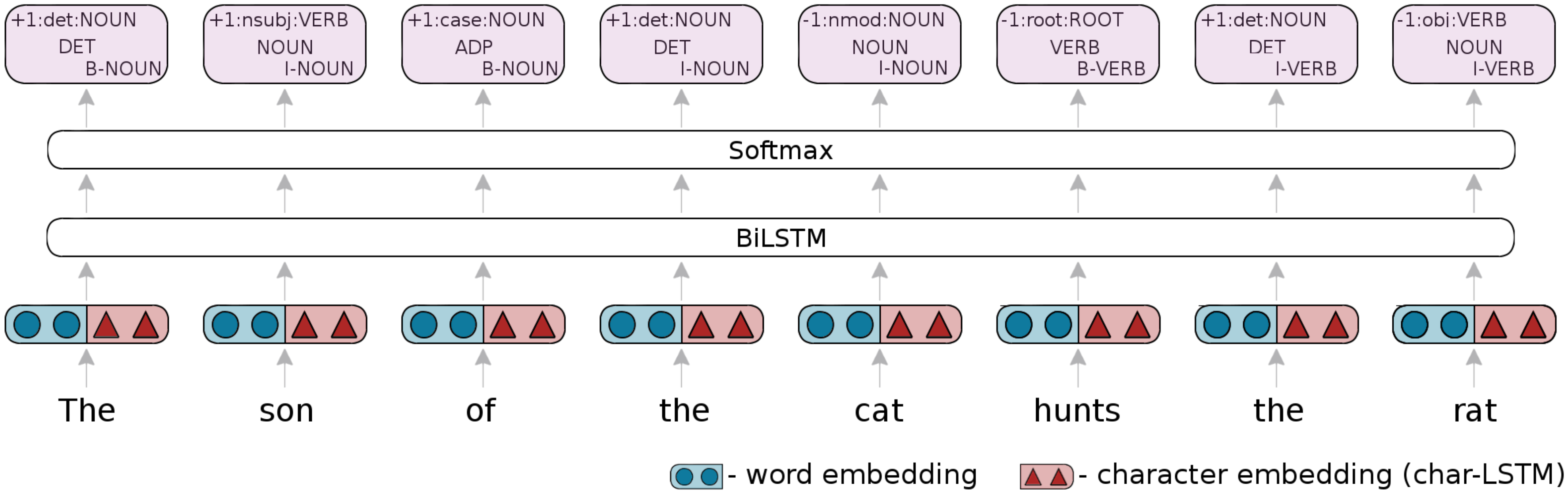
Japanse-GSD



System details.

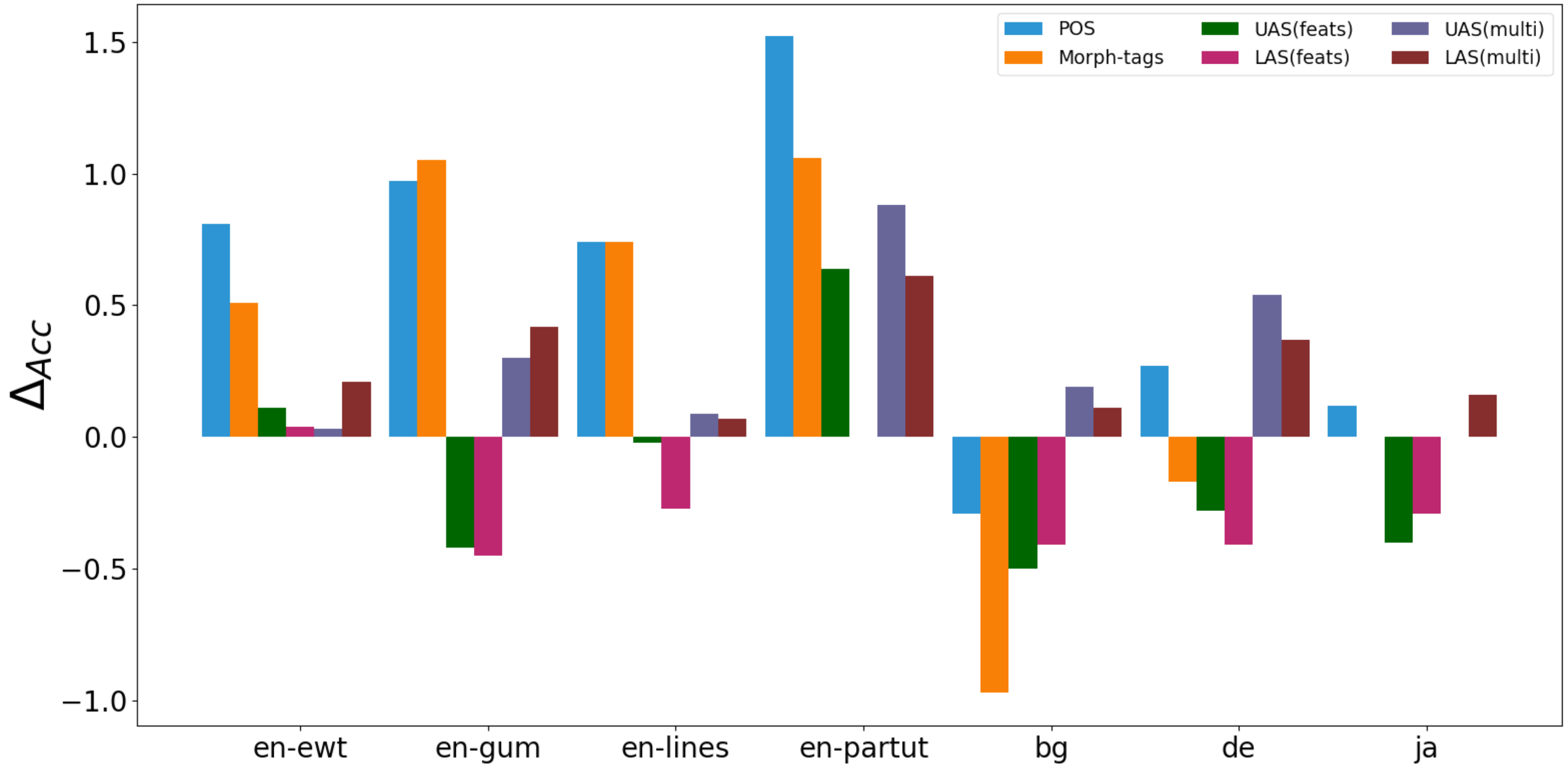
We use a neural sequence labelling toolkit, NCRF++.[1] And a relative POS tag position encoding for sequence labelling parsing.[2]

[1]Yang, J. and Zhang Y., *NCRF++: An Open-source Neural Sequence Labeling Toolkit*, 2018
[2]Spoustová, D.J. and Spousta M. *Dependency Parsing as a Sequence Labeling Task*, 2010



Broad results

Difference between **with** and **without** chunks.

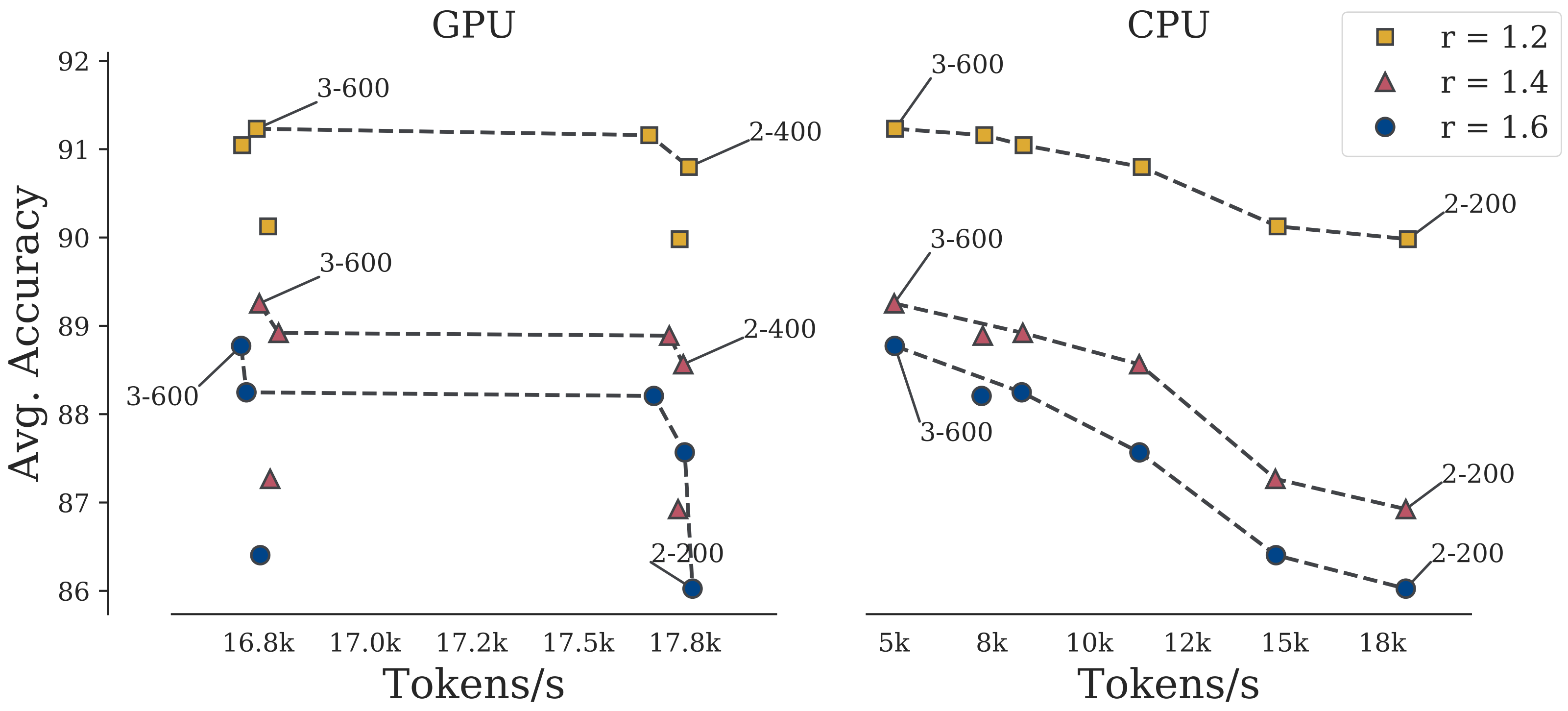


Chunk and Pass Parsing

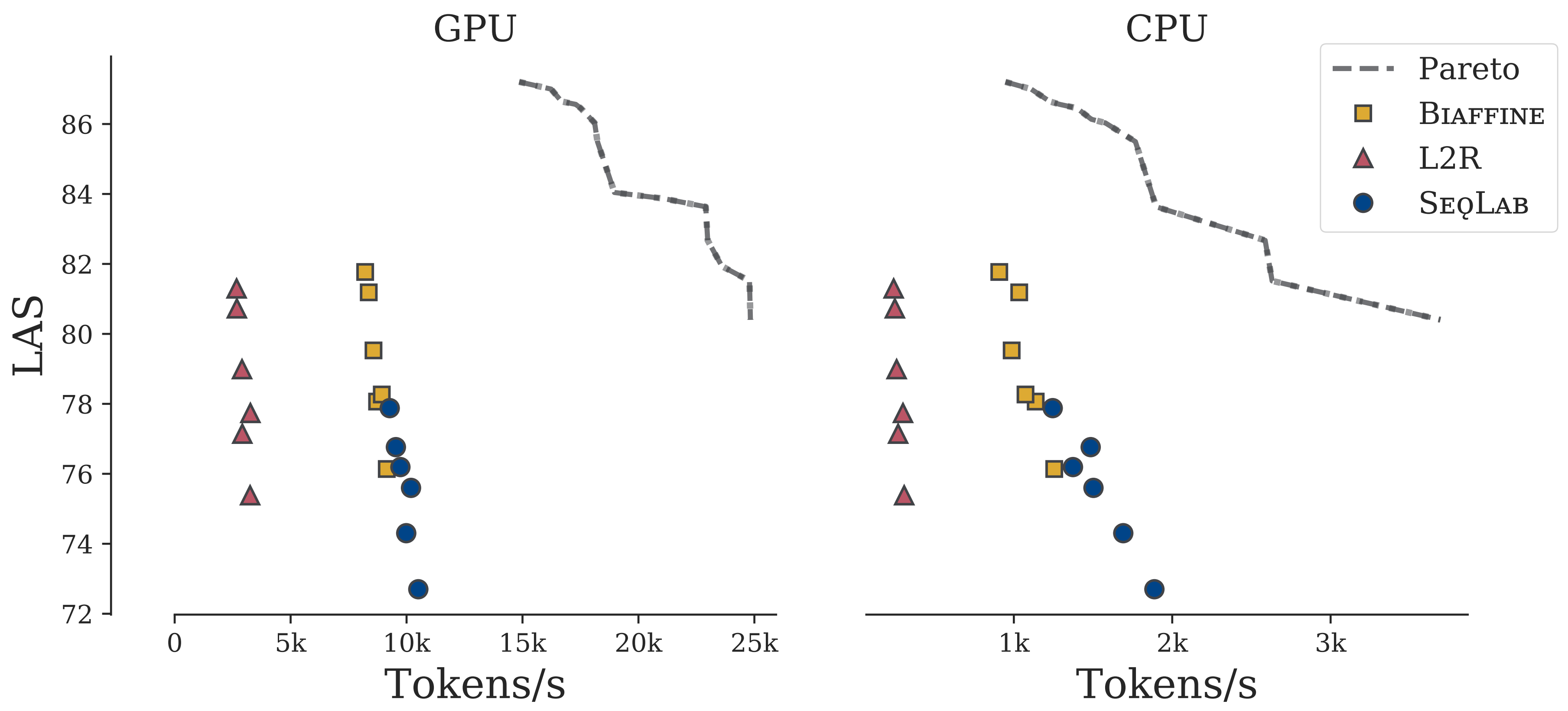
- Normalised PMI (used various thresholds to obtain different amount of compression)
- Compared to/with leading sytems (biaffine, l2r, and sequence labelling).
- All same network type (BiLSTM).

Chunker performance (BiLSTM w/ fastText and char embeddings)

Varying compression and network size. Reasonable performance.
Extends to labelled edge predictions.
(Chinese, Hindi, Korean, and Polish)

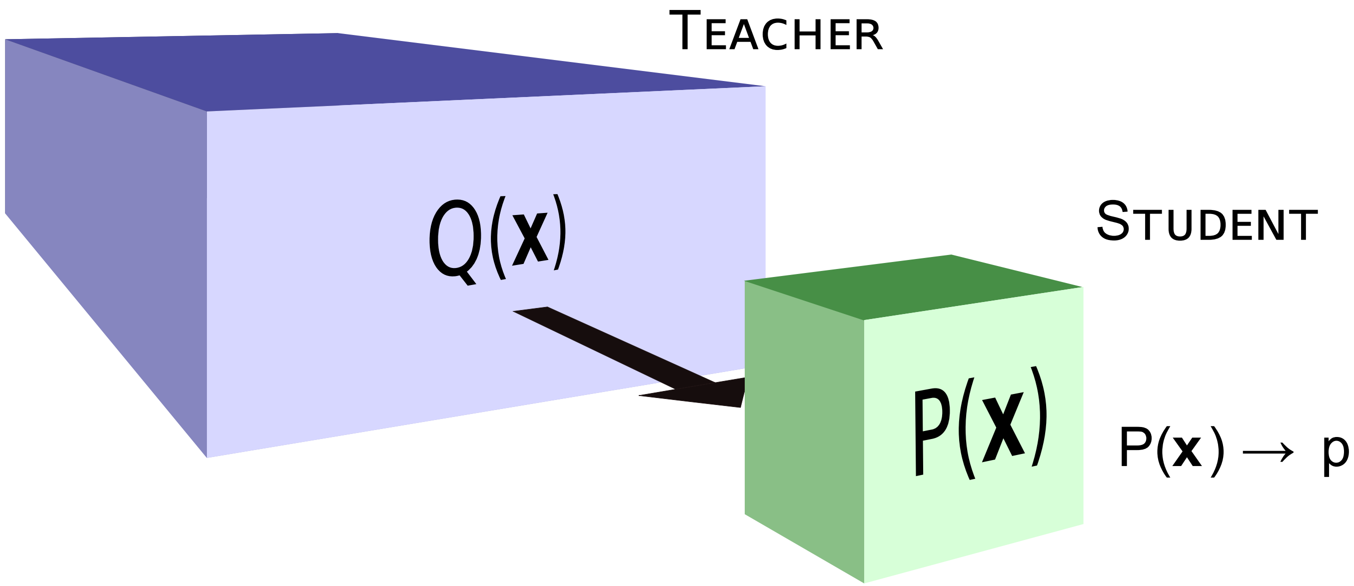


Pareto Front for UD (zh, hi, ko, pl)



Distillation

Teacher-student distillation



$P(\mathbf{x}) \rightarrow p(h|\,\mathbf{x})$

$\mathcal{L}_{KL}(Q(\mathbf{x}),P(\mathbf{x})) + \mathcal{L}_{CE}(p(h|\mathbf{x}))$

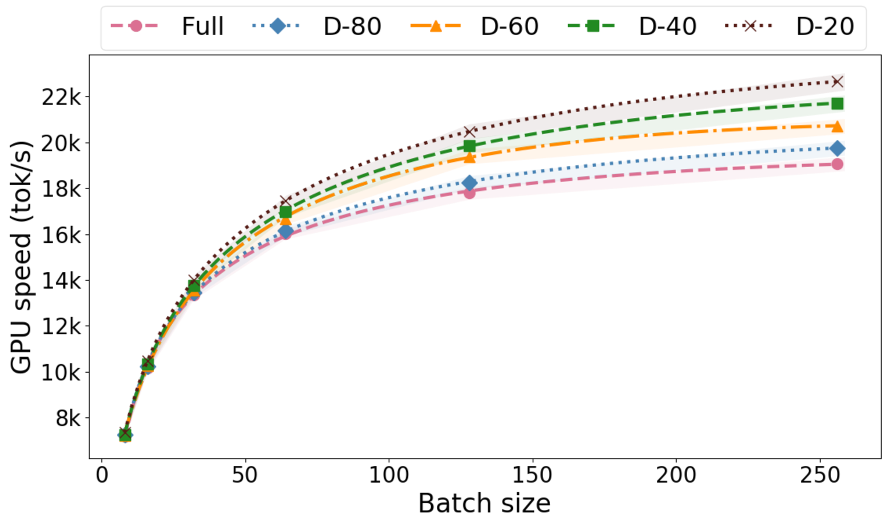
Experiments

- Used Biaffine (fast and accurate).
- Compared against teacher and small models.
- Used UDv2.4 (Ancient Greek, Chinese, English, Finnish, Hebrew, Russian, Tamil, Uyghur, and Wolof). Treebanks based on subset used in de Lhoneux et al. (2017).[1]
- Compress to 20%, 40%, 60%, and 80% of original model.
- fastText and gold POS tags. (°‸°)

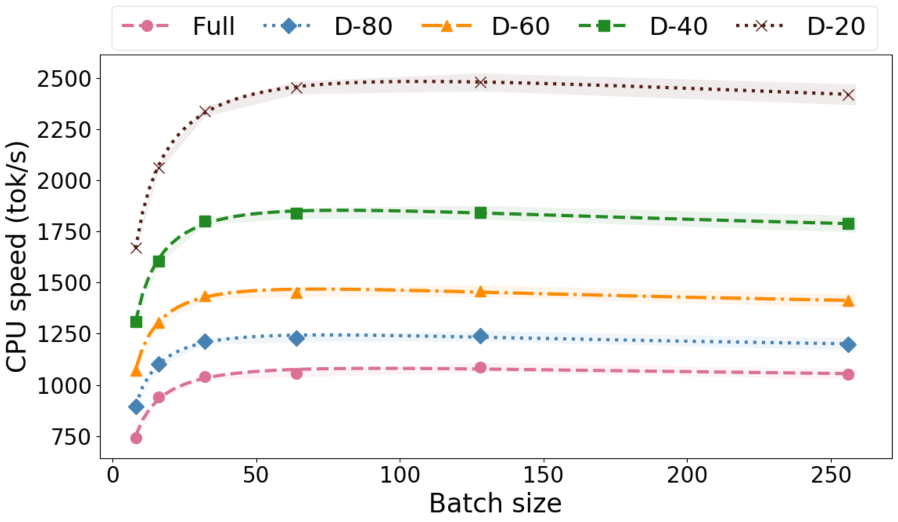
[1] de Lhoneux, M., Stymne, S. and Nivre, J. *Old school vs. new school: Comparing transition-based parsers with and without neural network enhancement*, 2017

Speed

GPU

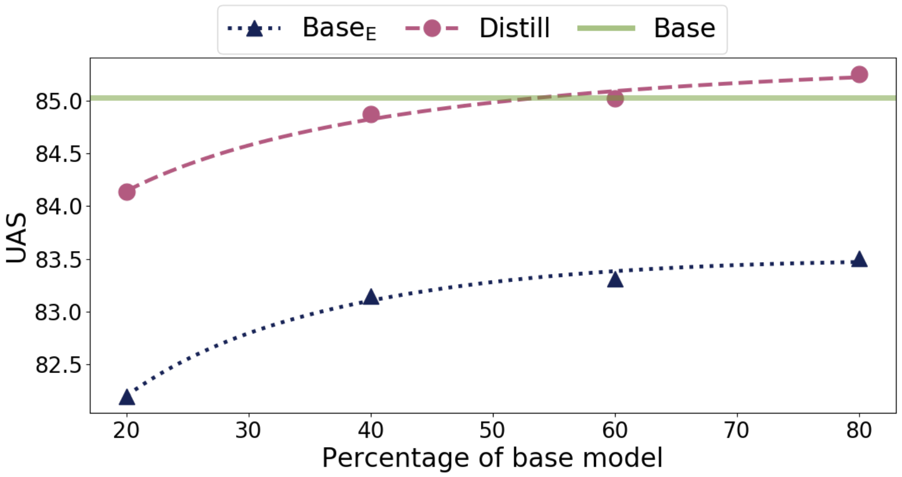


CPU

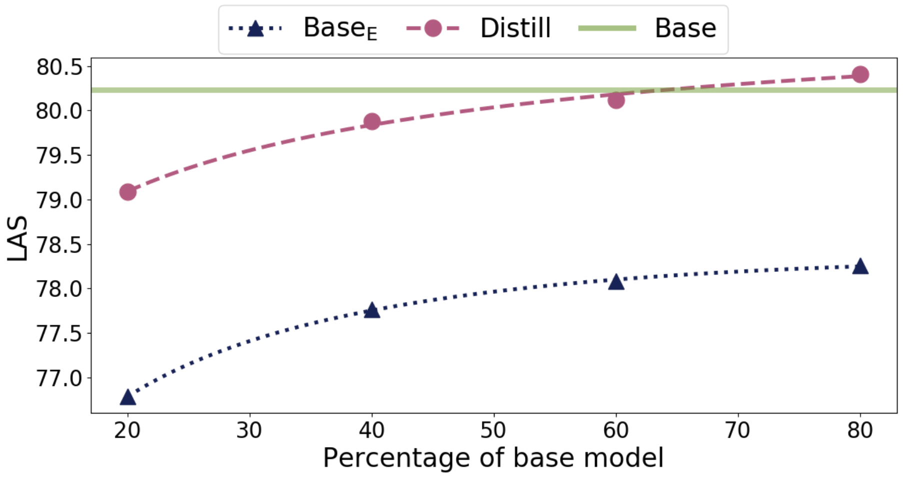


Accuracy

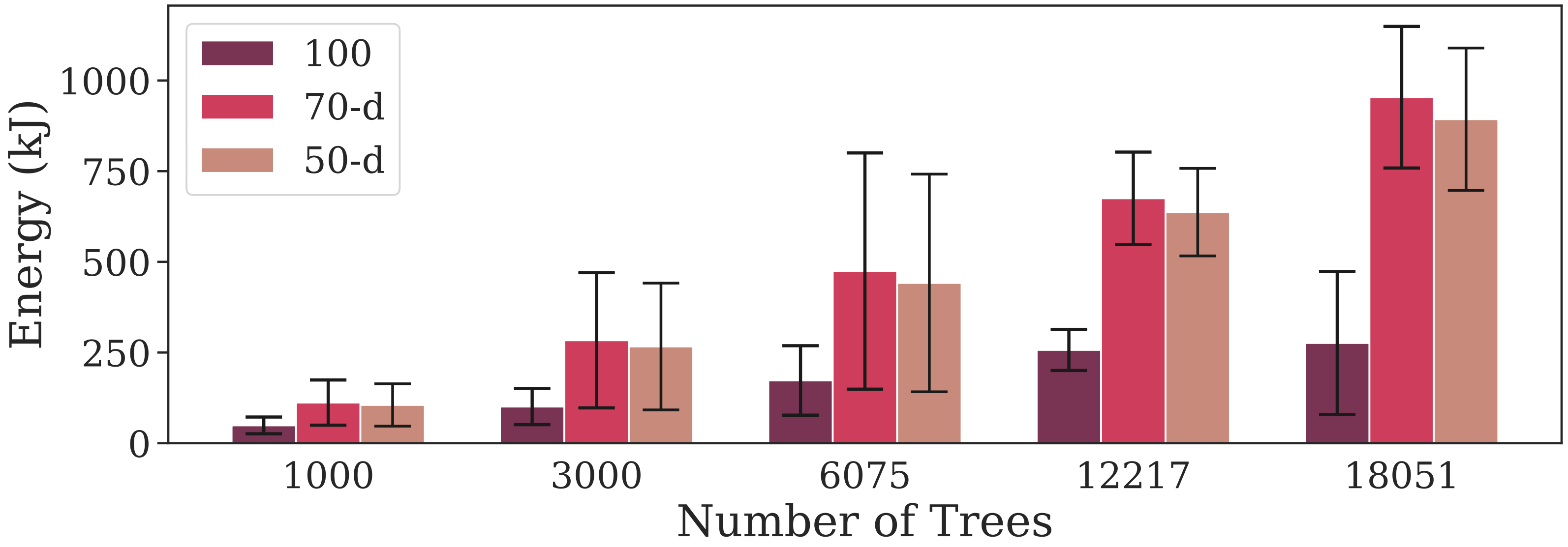
UAS



LAS



Energy cost

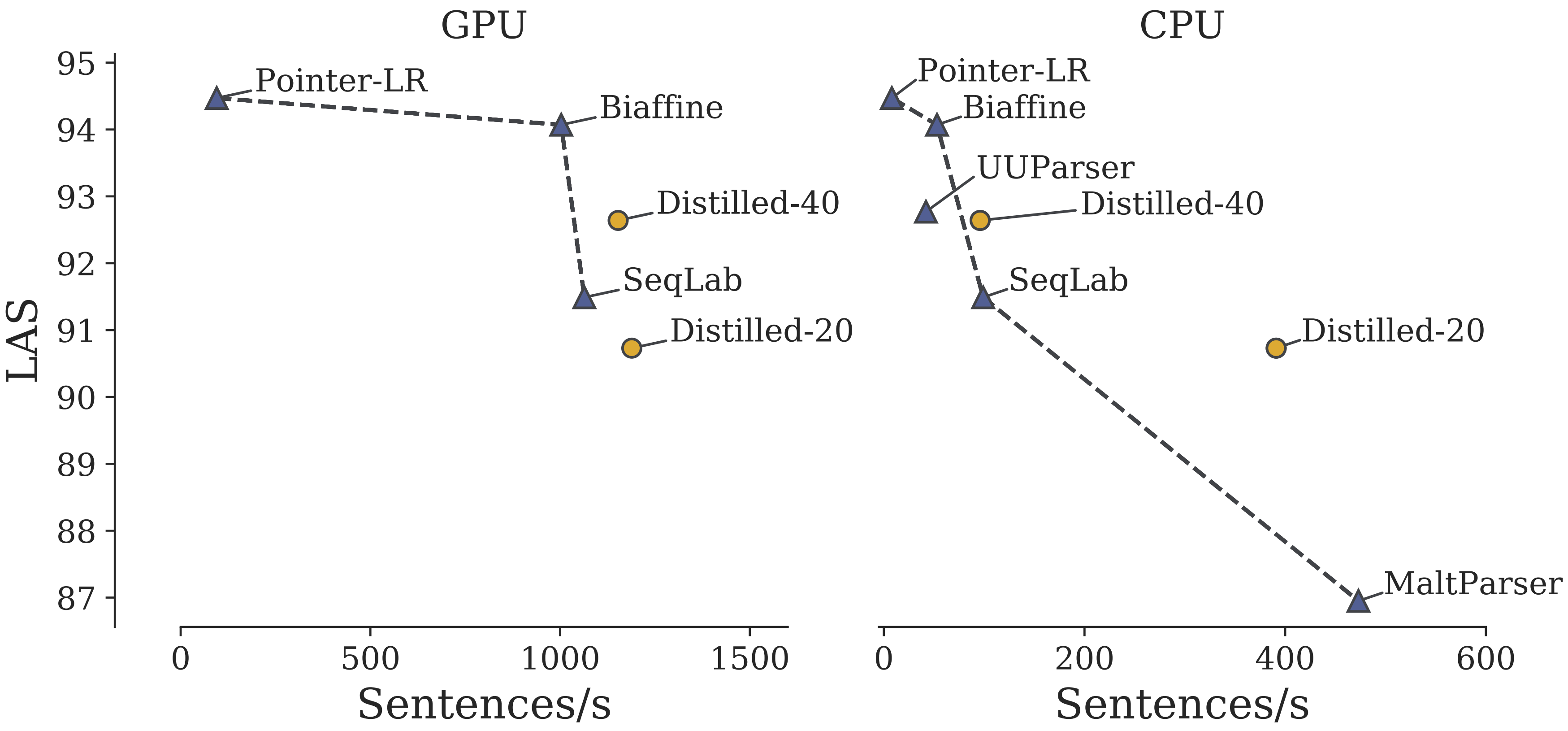


100
70-d
50-d
Energy (kJ)
0
250
500
750
1000
1000
3000
6075
12217
18051
Number of Trees

Inconsistent

- Subsequent work using only randomly initialised word and character.
- Didn't outperform small baselines.

Pareto Front for PTB (WSJ)



Part II

Evaluating Parsers

Dependency displacement

Transition-based algorithms

Different transition-based algorithms perform differently on different treebanks

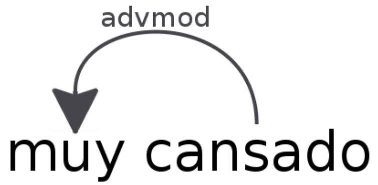
Perhaps because certain algorithms are inherently biased to creating edges that match given languages/treebanks?

- Used a non-NN parser, MaltParser.[1]
- Contains multiple algorithms.
- Differences between algorithms observed still seen in NN implementations.[2]
- 76 treebanks from UD v2.2.

[1] Nivre, J. et al., *MaltParser: A language-independent system for data-driven dependency parsing*, 2007.
[2] de Lhoneux, M., Stymne, S. and Nivre, J. *Old school vs. new school: Comparing transition-based parsers with and without neural network enhancement*, 2017

Transition-based algorithms

STACK	BUFFER
	Estoy muy ca..
SHIFT	b0 to top of STACK
Estoy	muy cansado
SHIFT	b0 to top of STACK
Estoy muy	cansado por...
REDUCE(right-advmod)	remove s0 from STACK
Estoy	cansado por...



Algorithms

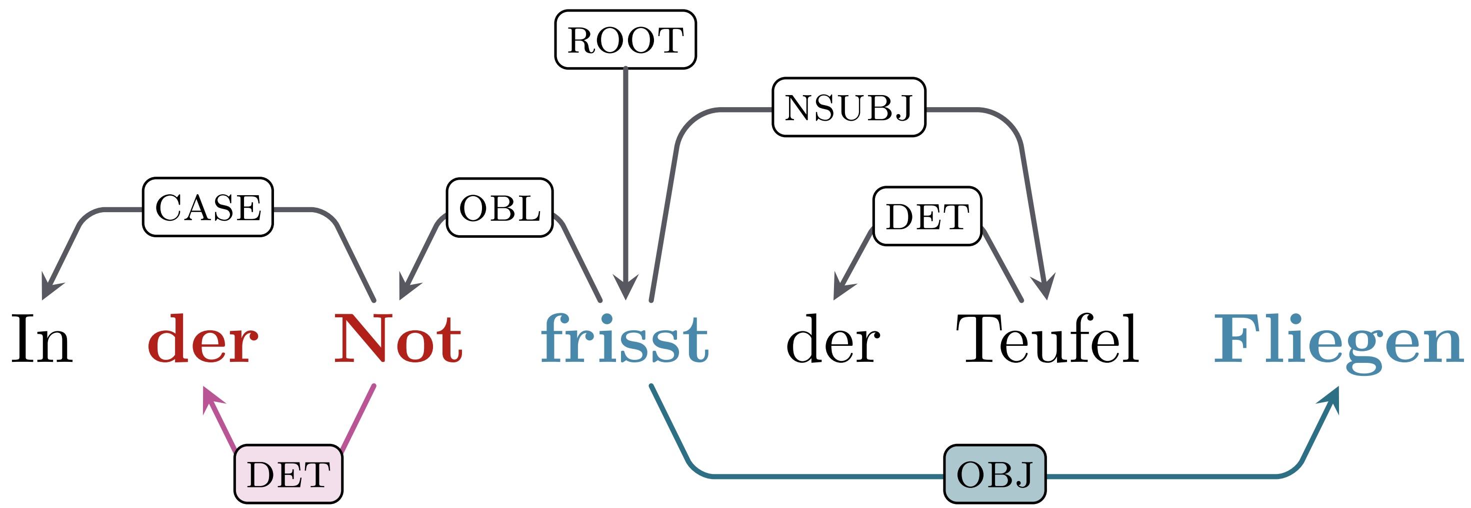
Projective

- Arc Standard
- Arc Eager
- Covington Projective

Non-projective

- Arc Swap
- Covington Non-projective

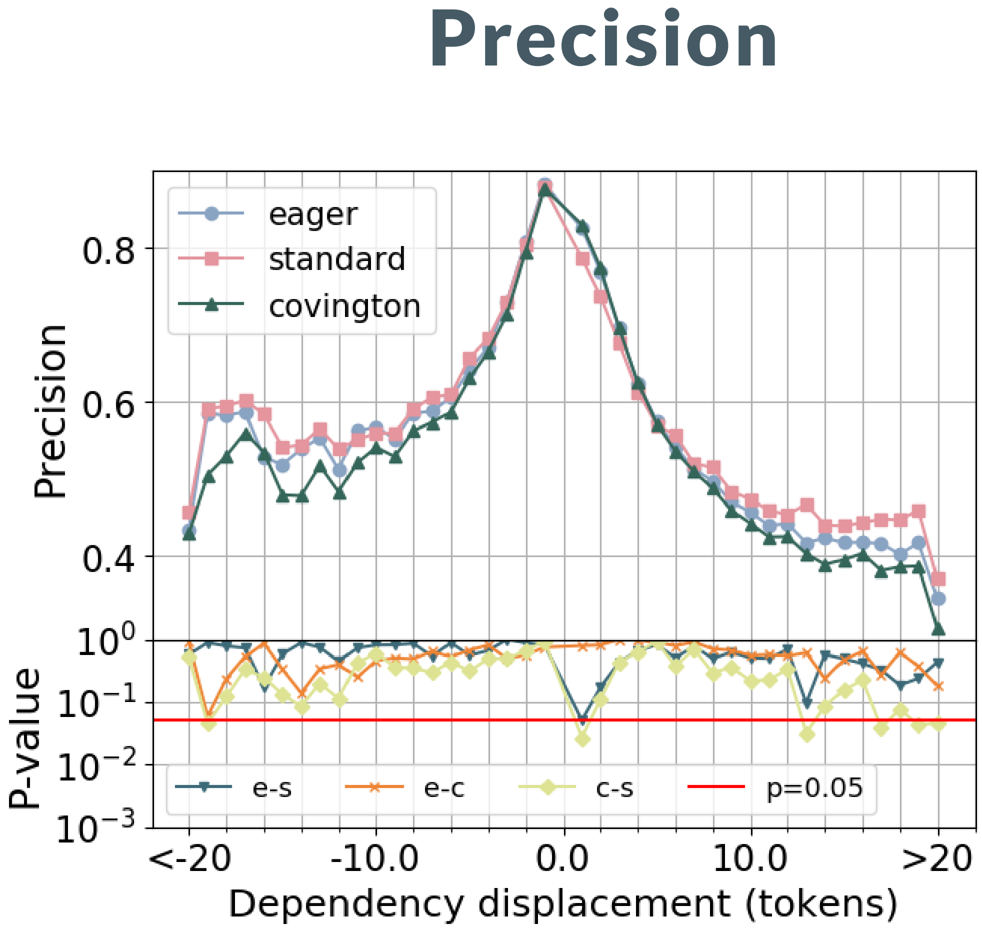
Dependency displacement

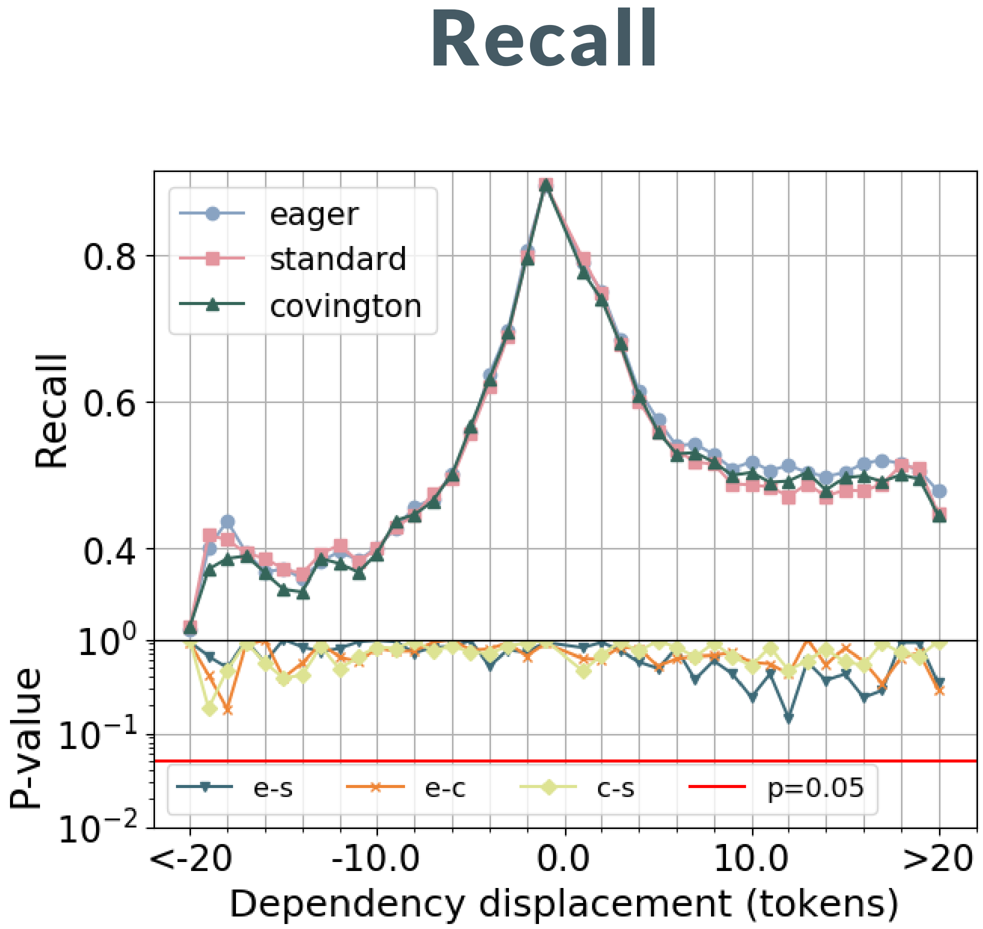


DET edge between der and Nott: -1 (2-3)

OBJ edge between frisst and Fliegen: 3 (7-4)

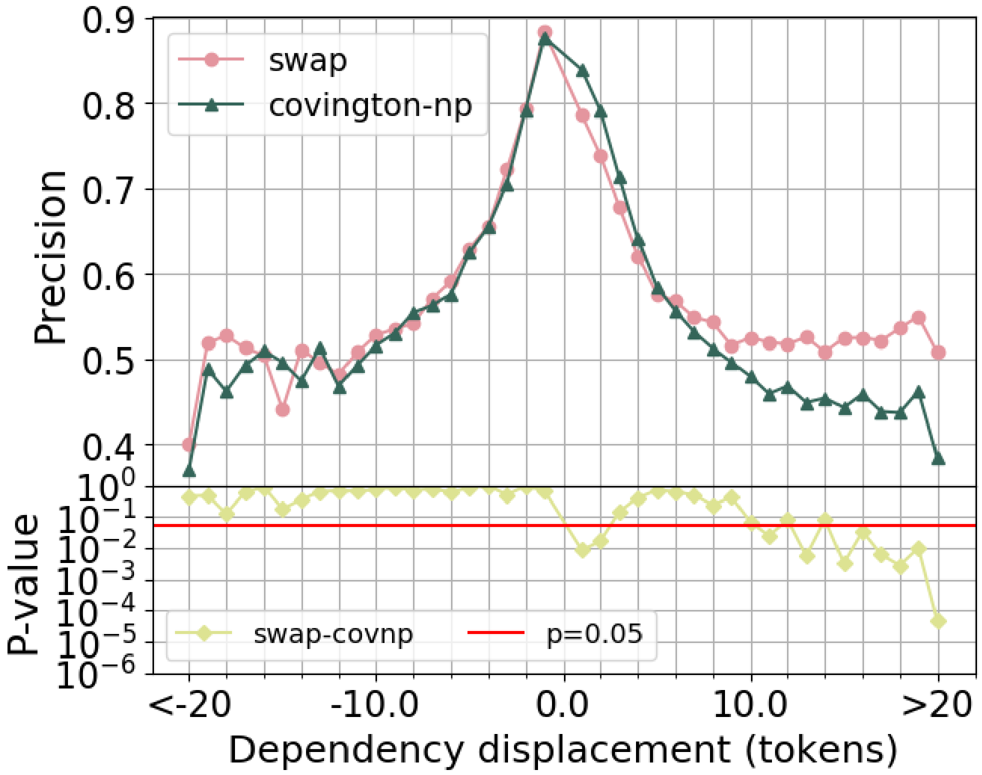
PROJECTIVE ALGORITHMS



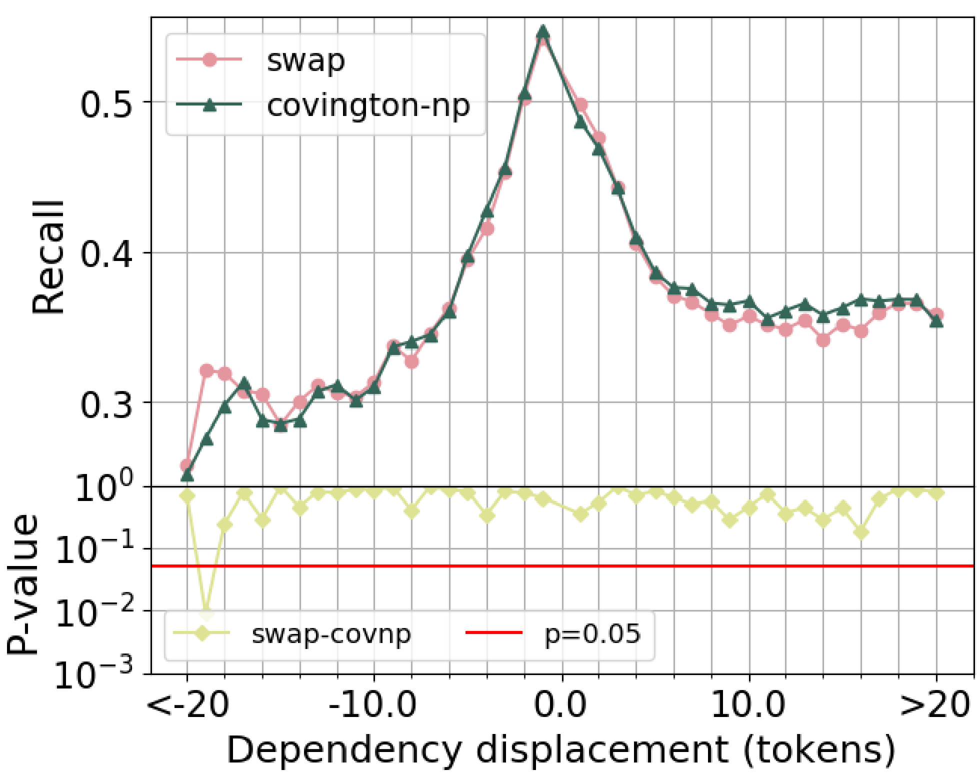


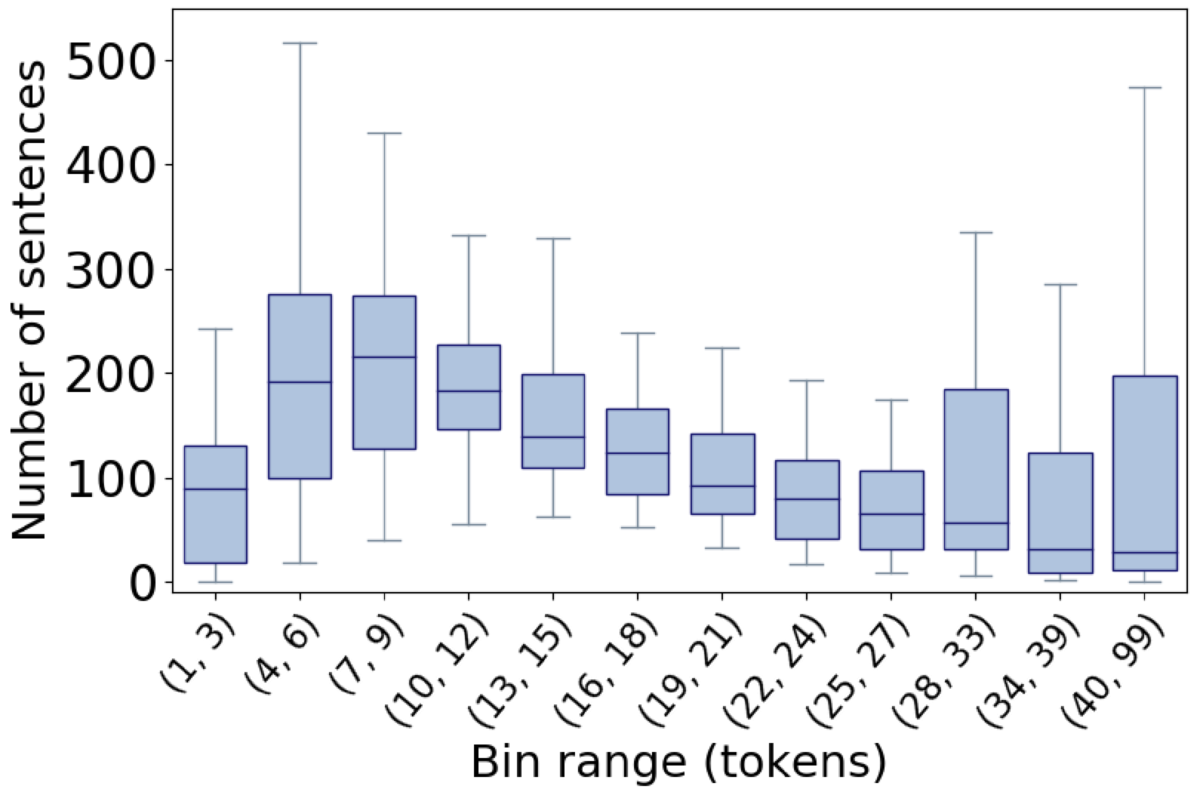
Non-projective algorithms

Precision

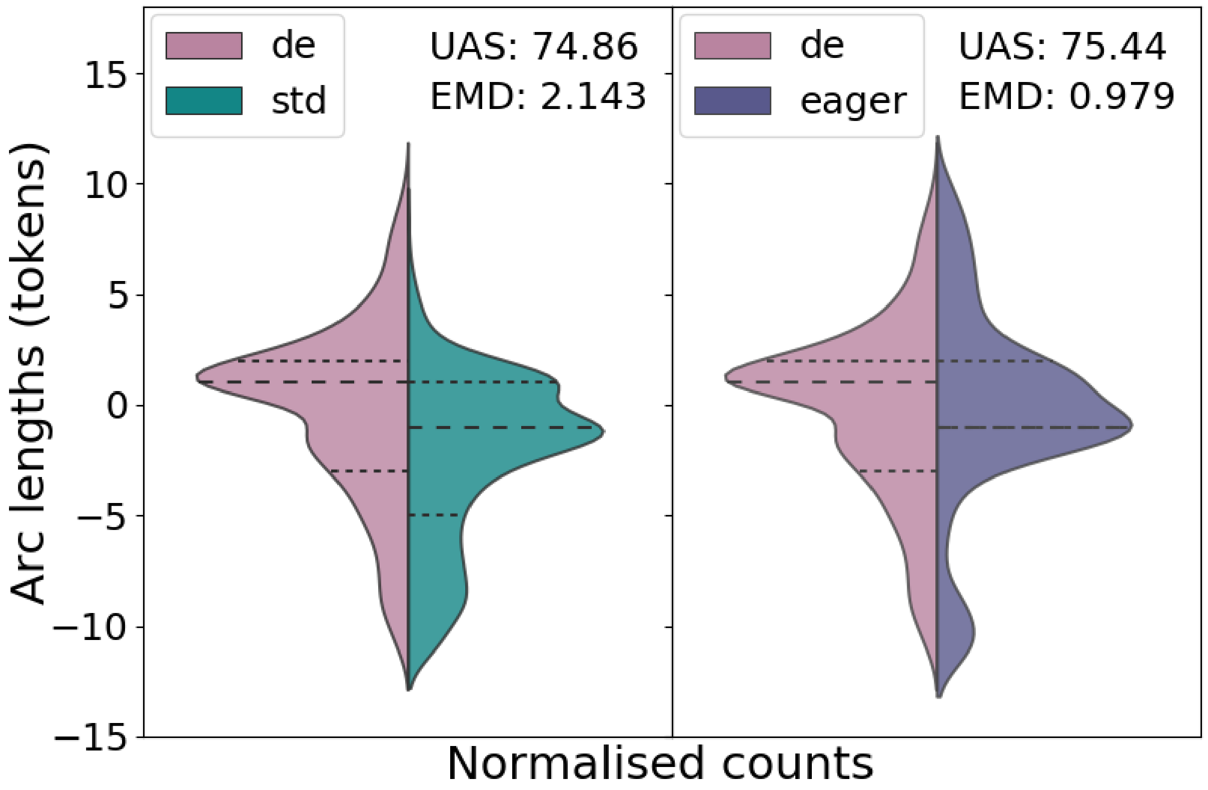


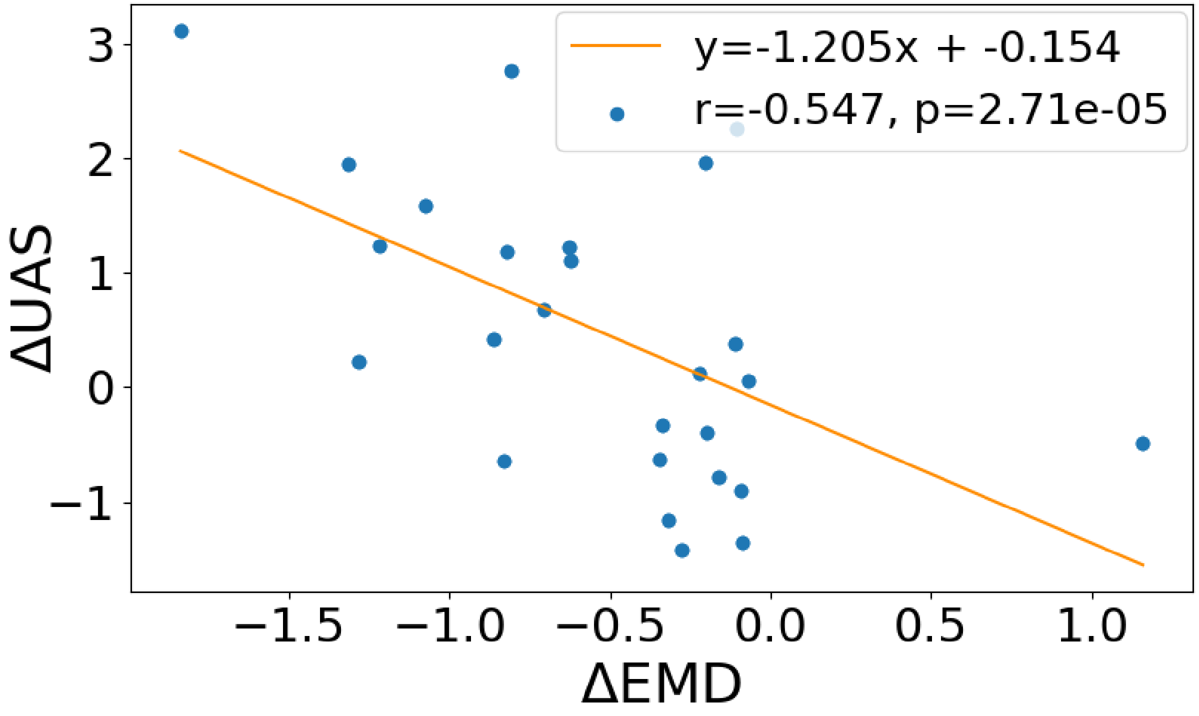
Recall

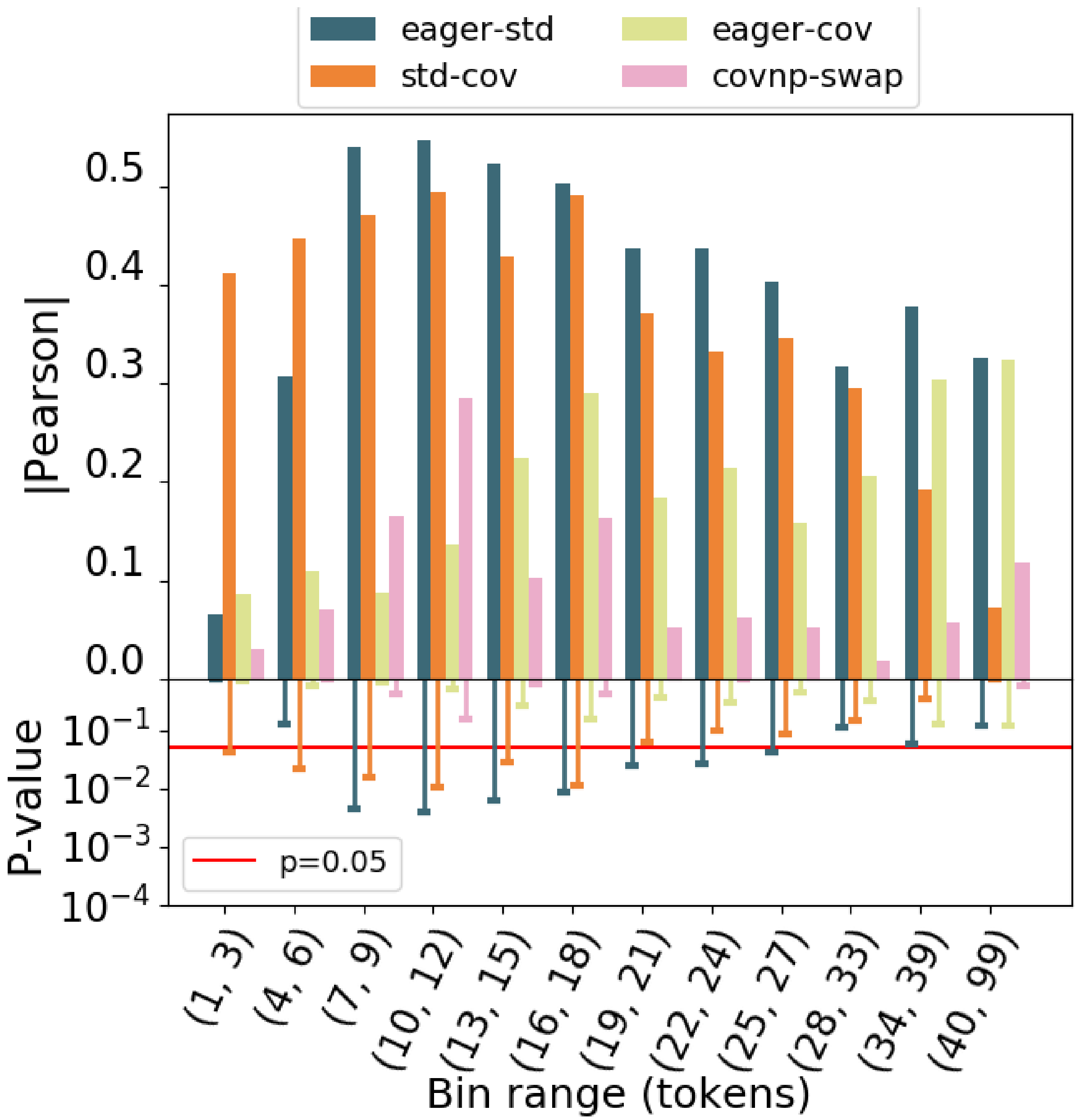




Number of sentences
0
100
200
300
400
500
(1, 3)
(4, 6)
(7, 9)
(10, 12)
(13, 15)
(16, 18)
(19, 21)
(22, 24)
(25, 27)
(28, 33)
(34, 39)
(40, 99)
Bin range (tokens)

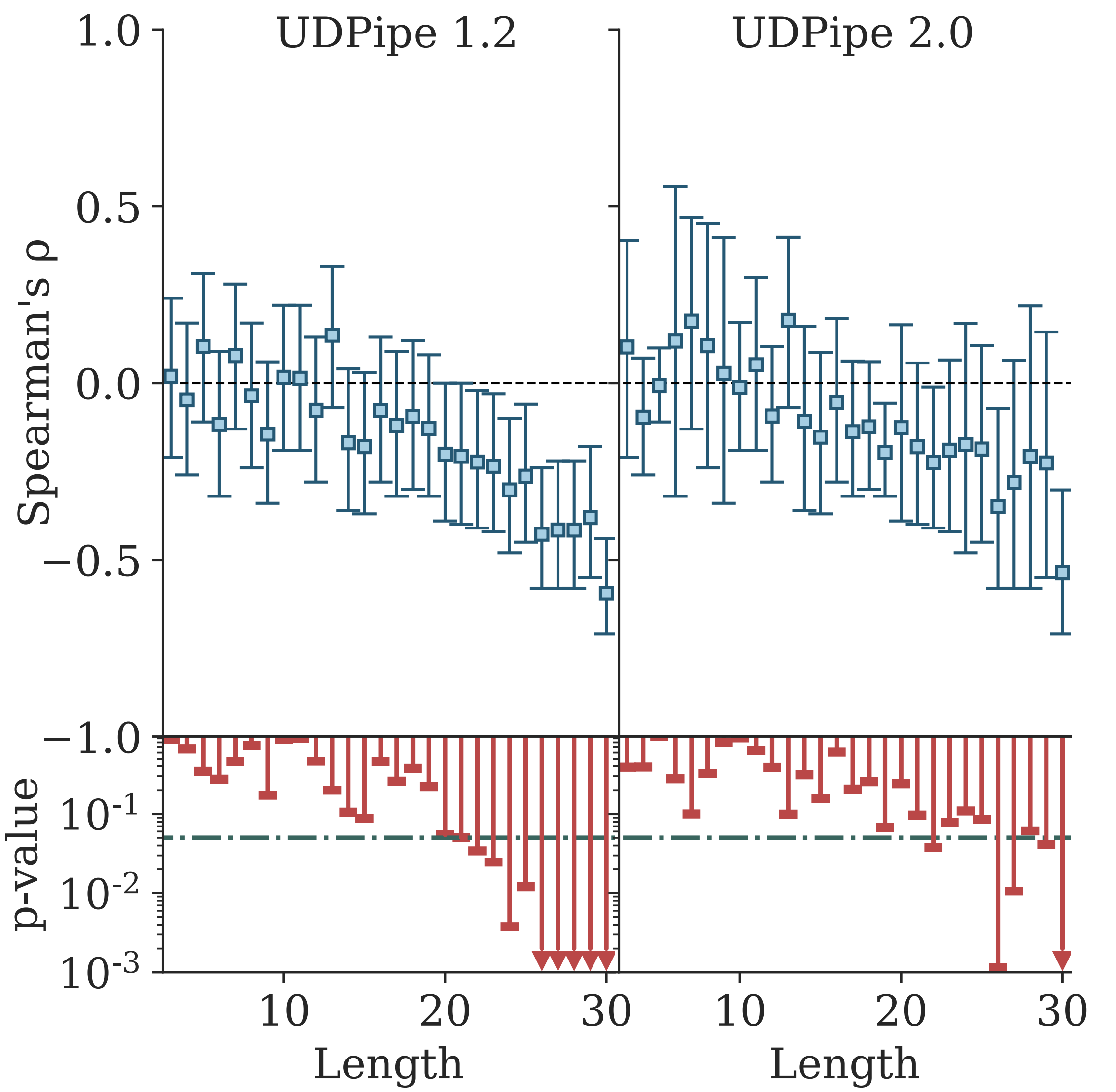

de
std
UAS: 74.86
EMD: 2.143
de
eager
UAS: 75.44
EMD: 0.979
Arc lengths (tokens)
15
10
5
0
−5
−10
−15
Normalised counts


y=-1.205x + -0.154
r=-0.547, p=2.71e-05
ΔUAS
ΔEMD
3
2
1
0
−1
−1.5
−1.0
−0.5
0.0
0.5
1.0

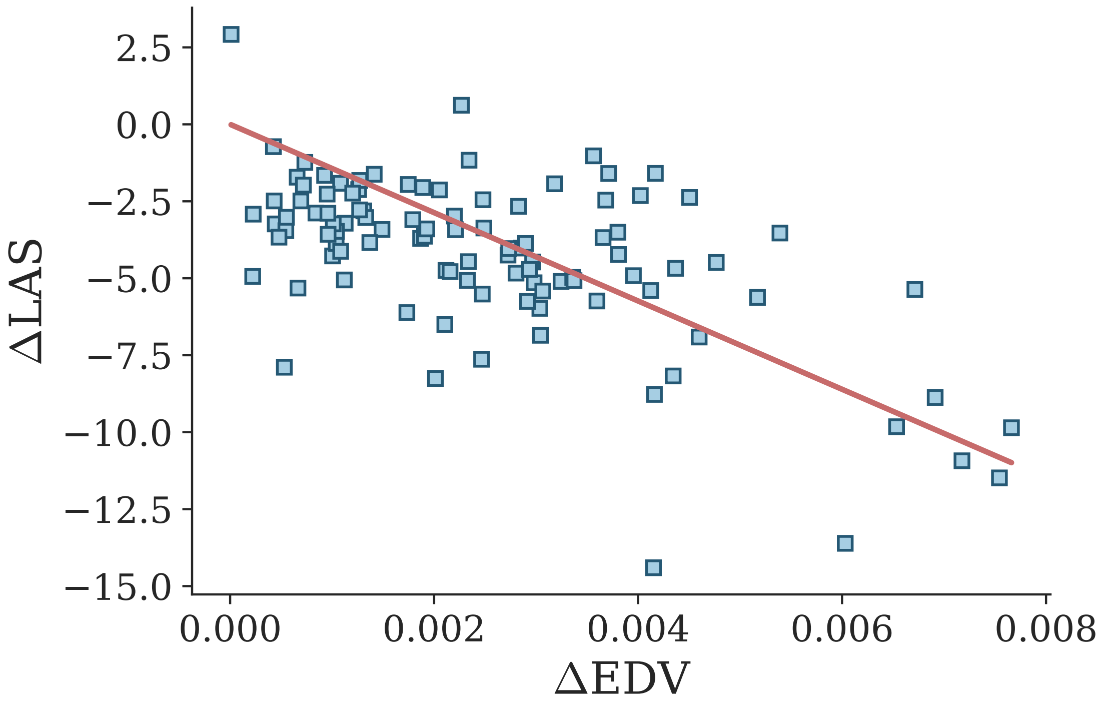


eager-std
std-cov
eager-cov
covnp-swap
|Pearson|
0.0
0.1
0.2
0.3
0.4
0.5
P-value
10^{-1}
10^{-2}
10^{-3}
10^{-4}
p=0.05
(1, 3)
(4, 6)
(7, 9)
(10, 12)
(13, 15)
(16, 18)
(19, 21)
(22, 24)
(25, 27)
(28, 33)
(34, 39)
(40, 99)
Bin range (tokens)

Training vs Test Data



Adversarial Splits



Universal POS tags

Parsers

UUParser

Transition-based parser out of Uppsala developed from TB BIST Parser.[1,2]

Biaffine

Graph-based parser developed from GB BIST Parser.[3]

word⊕char⊕upos

External pre-trained word embeddings, mainly fastText.

Same treebanks from distillation work.

[1]Kiperwasser, E. and Goldberg, Y. *Simple and accurate dependency parsing using bidirectional LSTM feature representations*, 2016

[2]Smith, A., de Lhoneux, M., Stymne, S. and Nivre, J. *An investigation of the interactions between pre-trained word embeddings, character models and POS tags in dependency parsing*, 2018

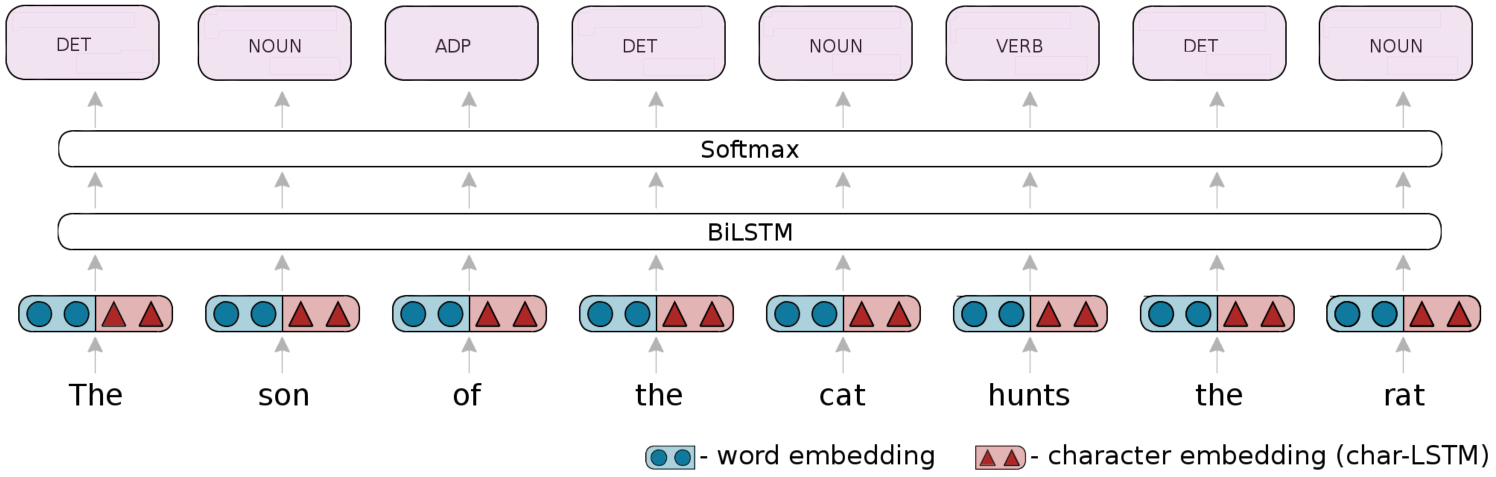
[3]Dozat, T. Manning, C.D., *Deep biaffine attention for neural dependency parsing*, 2017

Controlling UPOS accuracy

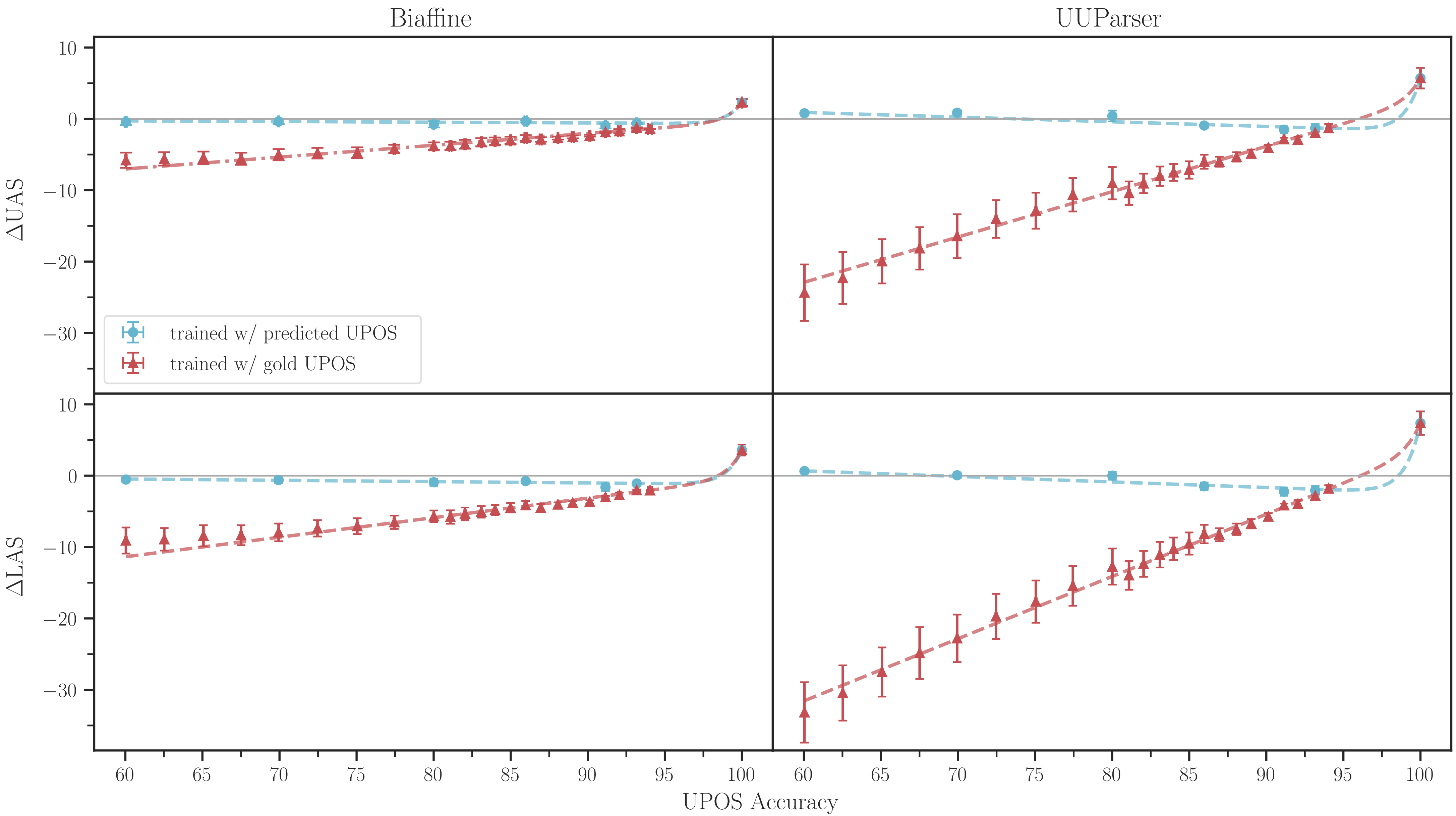
NCRF++ BiLSTM SL framework

BINS:: 2.5±0.3 from 60 to 80 and 1±0.3 from 80 onwards.

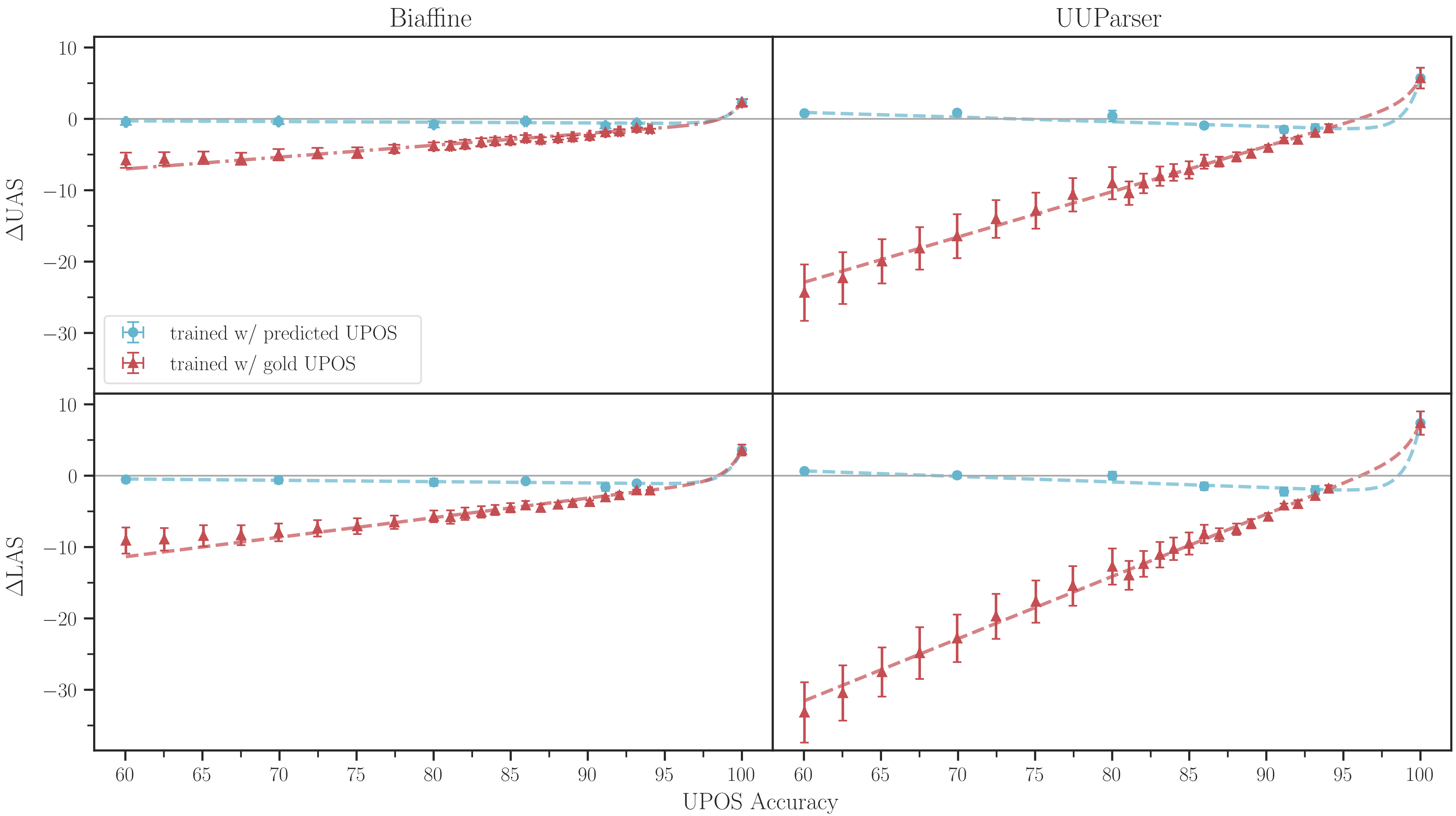
PARSERS: For training used gold tags and a subset of accuracy bins (60, 70, 80, 86, 91, and 93).



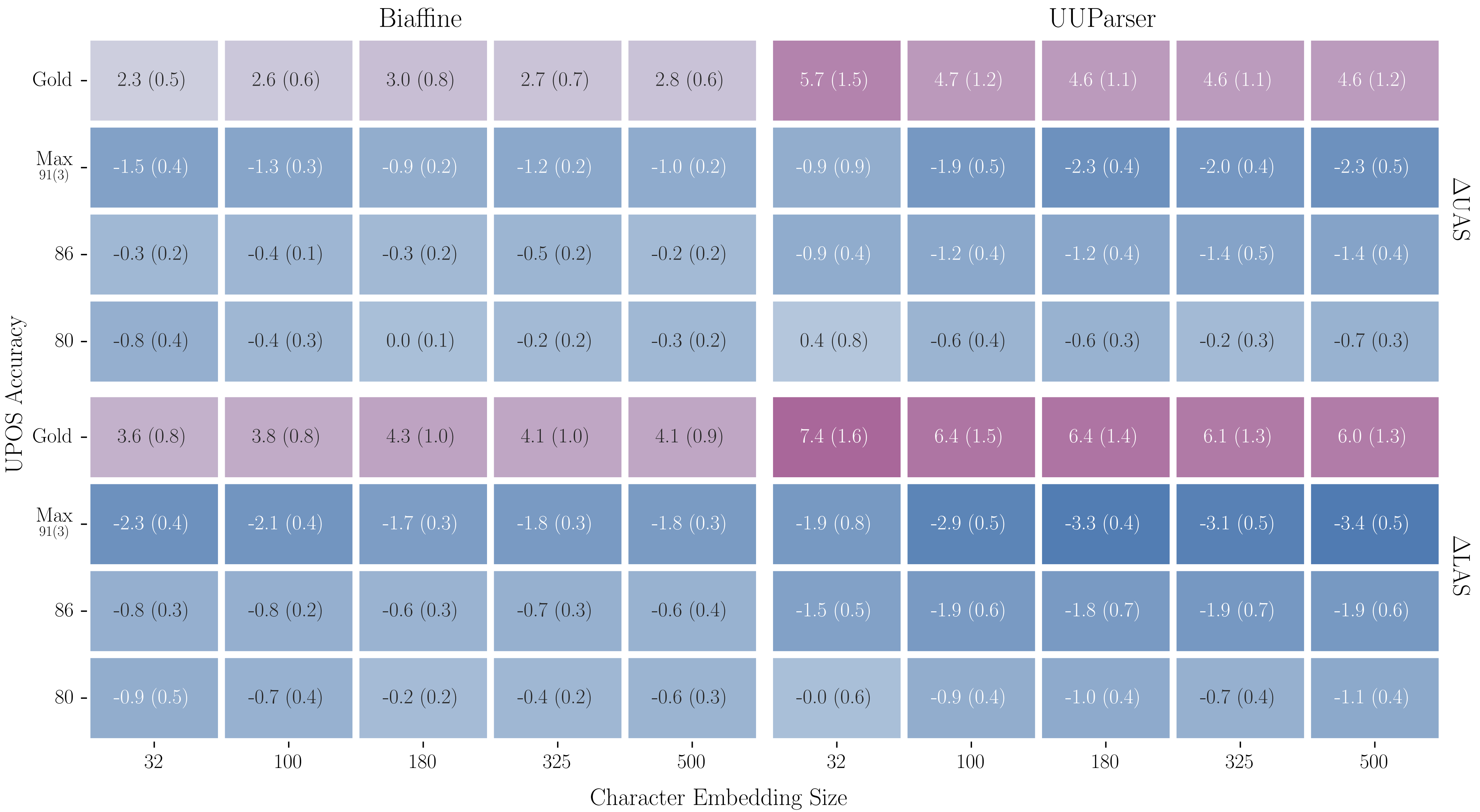
Experiment 1



Experiment 1

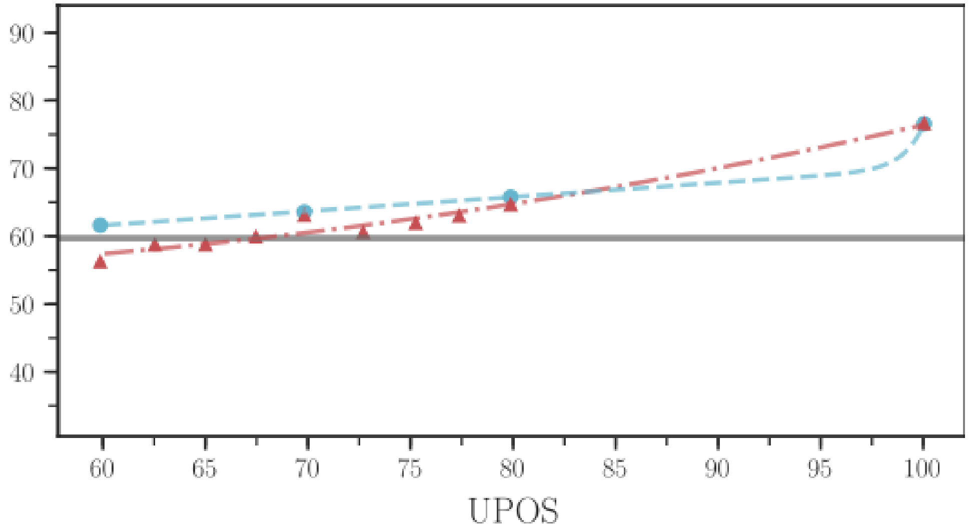


Experiment 2

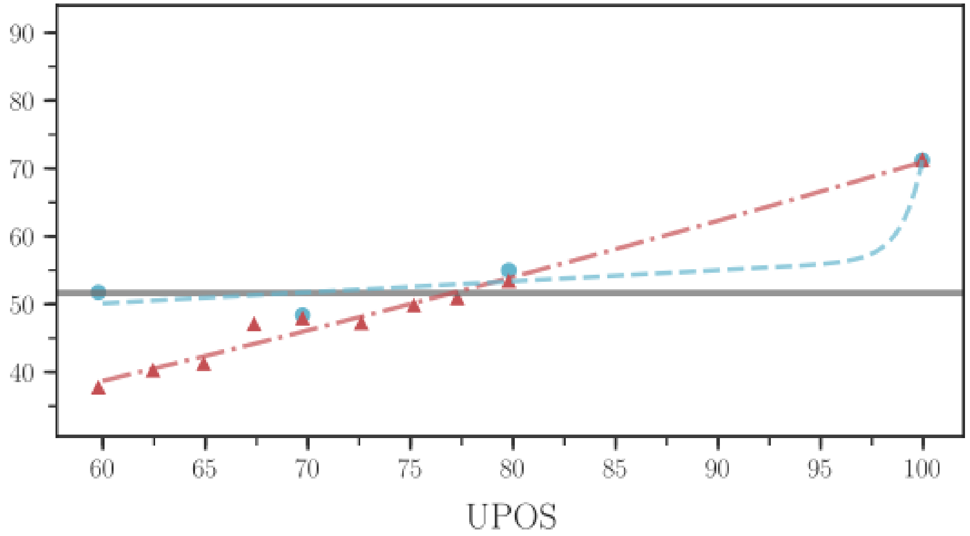


TAMIL RESULTS (~400 SENTENCES)

UAS



LAS



Still some improvement with low-accuracy taggers.

Very low-resource

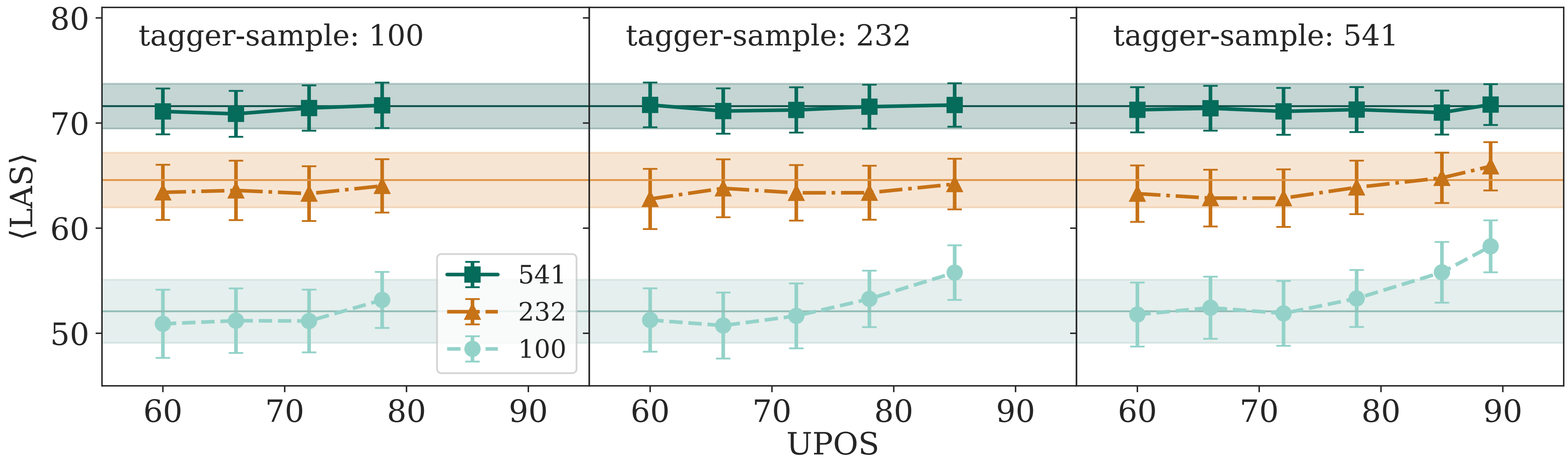
	UPOS		LAS			
	Single	Multi	None	Pred	Gold	Multi
bxr	48.72	48.34	10.45	12.36	20.31	14.41
kk	53.37	52.14	22.48	21.63	36.66	23.50
kmr	50.56	53.73	19.16	18.31	35.54	21.58
olo	37.84	37.37	9.74	10.89	17.54	7.59
hsb	53.44	47.28	18.36	20.03	41.88	14.66
avg	48.79	47.77	16.04	16.64	30.39	16.25

Fairly low-resourced

	UPOS		LAS			
	Single	Multi	None	Pred	Gold	Multi
be	92.82	87.29	61.82	64.91	68.87	62.28
gl	93.54	88.56	70.60	72.73	79.06	70.54
lt	79.25	71.51	37.17	35.94	48.30	38.96
mr	80.58	76.46	57.04	58.74	64.32	56.31
orv	87.77	81.60	49.53	51.34	60.24	50.33
ta	86.88	79.23	63.85	62.75	74.31	63.15
cy	91.77	86.41	72.10	72.93	80.71	73.00
avg	85.89	77.77	55.24	56.52	64.13	55.10

Artificial Low-resource

Indonesian GSD, Irish IDT, Japanese GSD, and Wolof WTB.

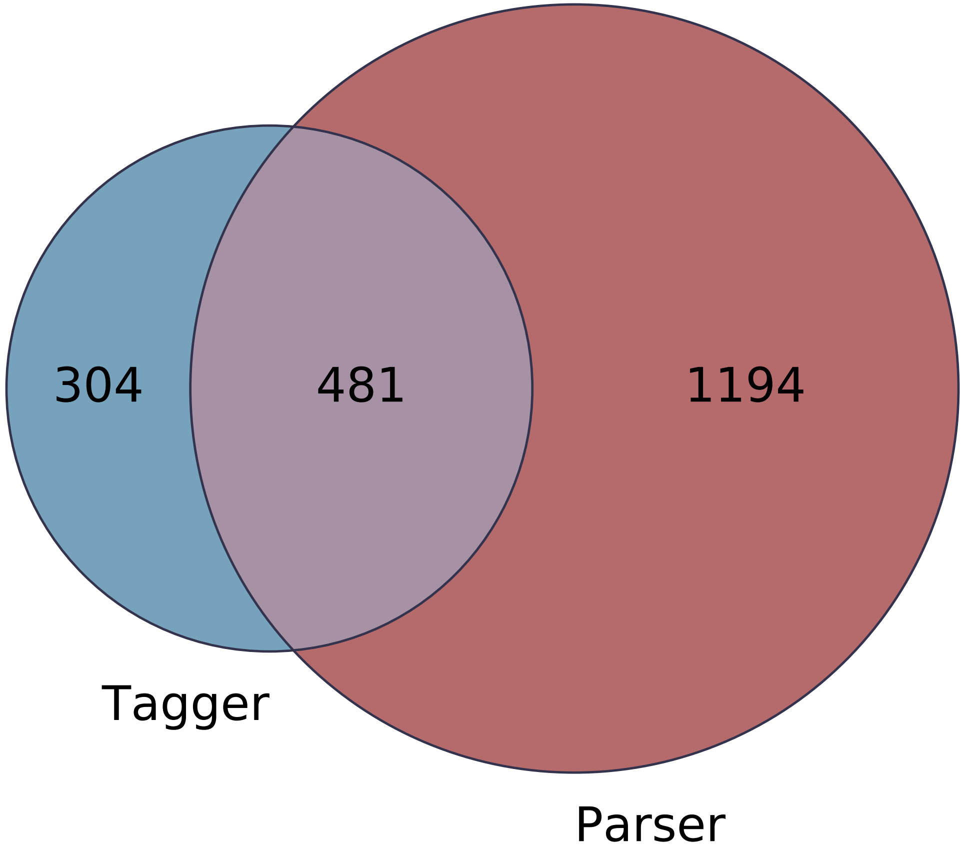


Probing Experiment

Arabic, Basque, Finnish, Indonesian, Irish, Japanese, Korean, Tamil, Turkish, Vietnamese, and Wolof

- Trained parser (Biaffine)
- Finetuned parser to predict POS tags
- Trained taggers with same sytem

Probing Experiment



Union of errors (average over treebanks).

Masking Experiment

	None	Pred.	$M\neg E_T$	$M\neg E_P$	$M\forall E_T$	Gold
ar	83.29	82.87	84.17	84.06	84.45	84.73
eu	81.12	81.14	82.33	82.62	83.13	84.45
fi	85.96	86.04	86.88	87.09	87.61	88.80
id	79.04	78.95	82.20	82.69	81.08	82.95
ga	76.13	76.57	76.62	76.65	77.46	77.90
ja	93.15	92.72	94.41	94.38	94.39	95.30
ko	85.40	85.86	87.53	87.82	87.44	88.52
ta	65.61	64.50	70.24	66.67	66.01	71.95
tr	66.67	67.68	67.62	67.66	67.84	68.86
vi	58.43	60.09	65.42	66.75	65.18	70.87
wo	77.87	78.49	82.03	81.39	81.11	85.41
avg	77.52	77.72	79.95	79.80	79.61	81.79

- None — no tags.
- Pred. — predicted tags.
- $M\neg E_T$ — gold tags except tagger errors.
- $M\neg E_P$ — gold tags except parser errors.
- $M\forall E_T$ — gold tags only tagger errors.
- Gold — all gold tags.

End

Conclusion

Developing

- Chunk-and-Pass
- Distillation

Evaluating

- Edge displacement
- POS tags

Collaborators (work presented)

Carlos Gómez Rodríguez

Mathieu Dehouck

David Vilares