Fast and Accurate Unlexicalized Parsing via Structural Annotations

Maximilian Schlund

joint work with
Michael Luttenberger and Javier Esparza

Chair of Theoretical Computer Science, TU München

http://xkcd.com/1090/
Unlexicalized PCFG-Parser from a Treebank

90% → training data → annotate → PCFG

10% → test data

baseline

treebank

parse test data

evaluate $LP/LR/F_1$

$LA/CB$
Tree Annotations – Plain PCFG

SIMPX

KOORD  VF

KON  NX

FKOORD

FKONJ  KON  FKONJ

MF  NX  NX

LK  VFIN  VXFIN

PDS  LK  MF

ADJA  ADJX  NN

PIAT  NN

Arbeit  bedeutet  Kosten

ist  viel

hohe

NX → \frac{1}{3} PDS

NX → \frac{1}{3} PIAT NN

NX → \frac{1}{3} ADJX NN
Kosten ist viel Arbeit bedeutet und das kostet viel Arbeit.
Tree Annotations – Height Annotation

SIMPX

KOORD   VF
KON   NX-2
Aber   PDS-1
das   LXFIN
VF   NX-2
VFIN   PIAT-1  NN-1
ist   viel   Arbeit
VF   VXFIN   NX-3
VFIN   ADJX-2  NN-1
bedeutet   ADJA   Kosten
hohe

NX-2  0.5 →  PDS-1
NX-2  0.5 →  PIAT-1  NN-1
NX-3  1 →  ADJX-2  NN-1
Tree Annotations – Dimension Annotation

SIMPX

KOORD  VF  FKOORD
KON     NX-0  FKOORD
Aber    PDS-0  KON
das     VXFIN  MF
ist     PIAT-0  NN-0
viel    Arbeit
bedeutet ADJA  Kosten
hohe

NX-0  \rightarrow  PDS-0
NX-1  \rightarrow  PIAT-0  NN-0
NX-1  \rightarrow  ADJX-0  NN-0
Tree Dimension, a.k.a. Strahler number

\[
\dim (\bullet) = 0 \quad \dim \left( \begin{array}{c}
  k \\
  < k
\end{array} \right) = k
\]

\[
\dim \left( \begin{array}{c}
  k \\
  k
\end{array} \right) = k + 1
\]
Tree Dimension II

Dimension = min. memory needed to traverse a (binary) tree

Height: 5, Dimension: 2.
Tree Dimension II

- Dimension = min. memory needed to traverse a (binary) tree

Height: 5, Dimension: 2
Tree Dimension II

Height: 5, Dimension: 2.

dimension = min. memory needed to traverse a (binary) tree
Tree Dimension II

Height: 5, Dimension: 2.

$\text{dimension} = \text{min. memory needed to traverse a (binary) tree}$
Tree Dimension II

Height: 5, Dimension: 2.
Tree Dimension II

Height: 5, Dimension: 2.

\[
\text{dimension} = \min \text{ memory needed to traverse a (binary) tree}
\]
Height: 5, Dimension: 2.
Tree Dimension II

Height: 5, Dimension: 2.

dimension = min. memory needed to traverse a (binary) tree
Dimension of Natural Language

Parse trees of natural language have small dimension ($\leq 4$)!
$\leadsto$ moderate blowup of $|G|$ when annotating.

This holds for all treebanks we looked at (Basque, English, French, Korean, ...)

German – TüBa-D/Z
Dimension of Natural Language

Parse trees of natural language have small dimension (≤ 4)! → moderate blowup of |G| when annotating.

This holds for all treebanks we looked at (Basque, English, French, Korean, ...)

French – SPMRL dataset
Experiments – Goals + Methods

- Evaluate all three annotation methods (Dim, Height, Parent) and combinations (Dim+Parent, Height+Parent).

- No fixed training/test set, 10 independent random samples of training/test data for studying variance, saved random seeds for reproducibility.

- Tools: python NLTK, Stanford parser

- Data: TüBa-D/Z treebank of written German (release 8.0; 75,000 trees)
### Results

| Annotation            | $|G|$ | Speed (sent./s) | PARSEVAL $F_1$ exact | Leaf-Ancestor LA (s) | LA (c) | Crossing brackets # CB | zero CB |
|-----------------------|-----|----------------|----------------------|-----------------------|--------|------------------------|---------|
| None                  | 21009 | 1.74 ± 0.04 | 84.8 24.4            | 84.0                  | 79.7   | 1.17                   | 58.5    |
| Parent                | 34192 | 1.07 ± 0.01 | 88.2 31.8            | 86.6                  | 82.9   | 1.07                   | 61.8    |
| Height                | 76096 | 3.06 ± 0.03 | 88.7 33.7            | 89.8                  | 86.2   | 0.93                   | 65.2    |
| Height+parent         | 130827 | 2.20 ± 0.04 | 89.2 36.8            | 90.8                  | 87.0   | 0.95                   | 65.4    |
| Dim                   | 49798 | **6.02 ± 0.10** | 88.5 31.8            | 89.7                  | 86.1   | 0.90                   | 64.9    |
| Dim+parent            | 84947 | 4.04 ± 0.07 | **90.4 39.1**        | **91.4**              | **88.1** | **0.85**            | **67.2** |

See [https://github.com/mschlund/nlp-newton](https://github.com/mschlund/nlp-newton) for all our scripts and data!
## Results

| Annotation       | $|G|$     | Speed (sent./s) | PARSEVAL $F_1$ exact | Leaf-Ancestor LA (s) LA (c) | Crossing brackets # CB zero CB |
|------------------|---------|-----------------|-----------------------|-----------------------------|--------------------------------|
| None             | 21009   | 1.74 ± 0.04     | 84.8 24.4             | 84.0 79.7                   | 1.17 58.5                      |
| Parent           | 34192   | 1.07 ± 0.01     | 88.2 31.8             | 86.6 82.9                   | 1.07 61.8                      |
| Height           | 76096   | 3.06 ± 0.03     | 88.7 33.7             | 89.8 86.2                   | 0.93 65.2                      |
| Height+parent    | 130827  | 2.20 ± 0.04     | 89.2 36.8             | 90.8 87.0                   | 0.95 65.4                      |
| **Dim**          | **49798** | **6.02 ± 0.10** | **88.5 31.8**         | **89.7 86.1**               | **0.90 64.9**                 |
| Dim+parent       | 84947   | 4.04 ± 0.07     | **90.4 39.1**         | **91.4 88.1**               | **0.85 67.2**                 |

**Dimension Annotation**

- $|G|$ grows by factor of 2.4.
- Parsing 3.5× faster than PCFG.
- $F_1$ gain comparable to parent annotation (+3%), LA scores much higher (+6%).

See [https://github.com/mschlund/nlp-newton](https://github.com/mschlund/nlp-newton) for all our scripts and data!
Results

| Annotation     | $|G|$   | Speed (sent./s) | PARSEVAL $F_1$ exact | Leaf-Ancestor LA (s) | LA (c) | Crossing brackets |
|----------------|-------|-----------------|----------------------|-----------------------|--------|-------------------|
| None           | 21009 | 1.74 ± 0.04     | 84.8                 | 24.4                  | 84.0   | 79.7              |
| Parent         | 34192 | 1.07 ± 0.01     | 88.2                 | 31.8                  | 86.6   | 82.9              |
| Height         | 76096 | 3.06 ± 0.03     | 88.7                 | 33.7                  | 89.8   | 86.2              |
| Height+parent  | 130827| 2.20 ± 0.04     | 89.2                 | 36.8                  | 90.8   | 87.0              |
| Dim            | 49798 | 6.02 ± 0.10     | 88.5                 | 31.8                  | 89.7   | 86.1              |
| Dim+parent     | 84947 | 4.04 ± 0.07     | 90.4                 | 39.1                  | 91.4   | 88.1              |

Dimension Annotation

- $|G|$ grows by factor of 2.4.
- Parsing 3.5× faster than PCFG.
- $F_1$ gain comparable to parent annotation (+3%), LA scores much higher (+6%).

Dimension + Parent Annotation

- $|G|$ grows by factor of 4.0.
- Parsing 2.3× faster than PCFG.
- Accuracy gains almost add (+5% $F_1$, +7% LA)

See https://github.com/mschlund/nlp-newton for all our scripts and data!
Summary + Current/Future Work

- Dimension as complexity measure for parse trees.
- Dimension/Height/Parent annotation, comparison.
- Dim.+Parent annotation improves parsing (speed + acc.)
Summary + Current/Future Work

- **Dimension** as complexity measure for parse trees.
- Dimension/Height/Parent annotation, comparison.
- Dim.+Parent annotation improves parsing *(speed + acc.)*

TODO ...

- We would love to **collaborate** with NLP-experts!
- Integrate strategies into parsing frameworks.
- Horizontal markovization, state-merging, morphological features in POS-tags
- Study other languages *(morphologically rich languages?)*.
- Explain parsing speedup theoretically.
- Look for further applications: dependency parsing (?)

Thank you!
Summary + Current/Future Work

- **Dimension** as complexity measure for parse trees.
- Dimension/Height/Parent annotation, comparison.
- Dim.+Parent annotation improves parsing (**speed + acc.**)

TODO ... 

- We would love to **collaborate** with NLP-experts!
- Integrate strategies into parsing frameworks.
- Horizontal markovization, state-merging, morphological features in POS-tags
- Study other languages (morphologically rich languages?).
- Explain parsing speedup theoretically.
- Look for further applications: dependency parsing (?)

Thank you!