Generalized Chart Constraints for Efficient PCFG and TAG Parsing

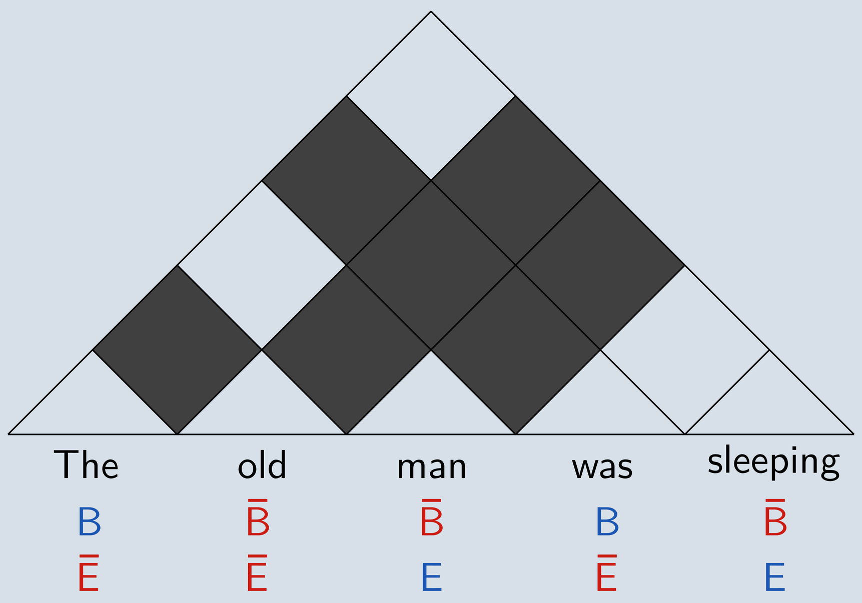
Stefan Grünewald, Sophie Henning, and Alexander Koller
Department of Language Science and Technology, Saarland University

Overview

- Roark et al. (2012) show that **chart constraints** are a simple and effective way to boost both efficiency and accuracy of PCFG chart parsing.

Chart constraints: Basic idea

- Use a tagger to predict at which string positions multi-word constituents may begin or end.
- Use these predictions to close off chart cells for disallowed spans:



$\bar{B} = \{1, 2, 4\}$
$\bar{E} = \{0, 1, 3\}$

- **Contribution 1:** We generalize chart constraints for use with other grammar formalisms, such as TAG.
- **Contribution 2:** We combine chart constraints with other pruning mechanisms and achieve speedups of up to 70x for PCFG and 124x for TAG, with no loss in accuracy.

Generalized chart constraints

- For many grammar formalisms, the parsing process can be expressed in terms of **parsing schemata** (Shieber at al., 1995), e.g. in the case of PCFG:

$$\frac{[B, i, j] \quad [C, j, k] \quad A \to B\ C}{[A, i, k]}$$

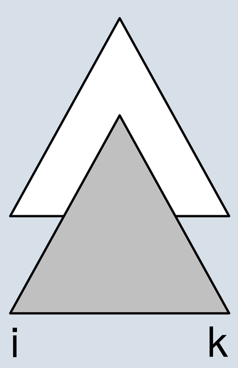
- Given a parsing schema, chart constraints can be interpreted as a set Q of **allowable parse items**:

$$\frac{[B, i, j] \quad [C, j, k] \quad A \to B\ C \quad [A, i, k] \in Q}{[A, i, k]}$$

Allowable parse items for PCFG

- Parse items $[A, i, k]$ for PCFG encode that the substring from i to k can be derived from the nonterminal A.
- For a parse item to be allowable, the span must obey the chart constraints:

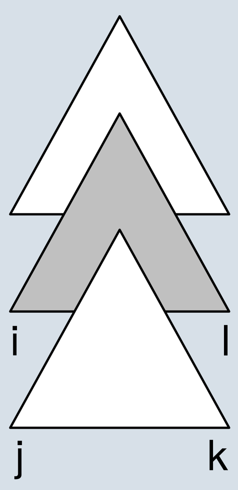
$$i \notin \bar{B} \;\wedge\; k \notin \bar{E}$$



Allowable parse items for TAG

- Parse items $[\mathcal{X}, i, j, k, l]$ for TAG additionally encode "gaps" (j, k) at which auxiliary trees may be adjoined.
- These gaps must obey the chart constraints as well:

$$i \notin \bar{B} \;\wedge\; j \notin \bar{B} \;\wedge\; k \notin \bar{E} \;\wedge\; l \notin \bar{E}$$

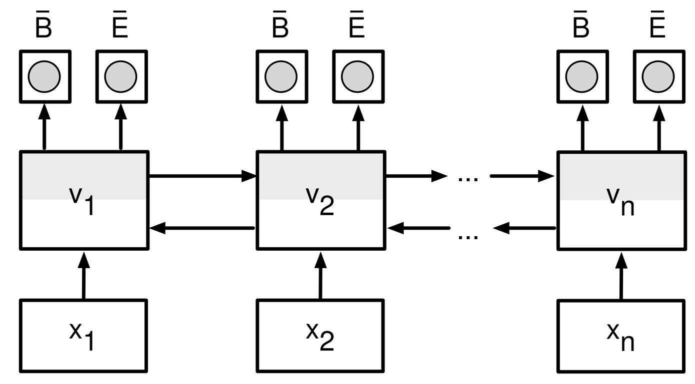


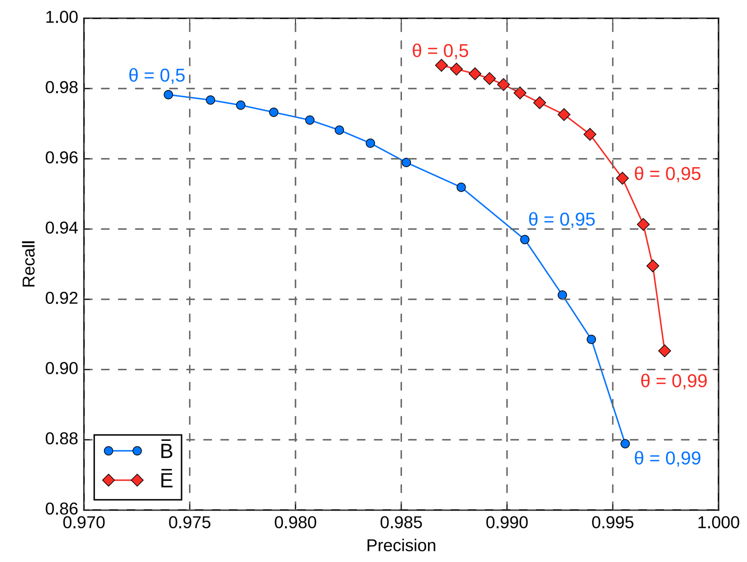
Predicting chart constraints

- We predict the probabilities of chart constraints at each string position using a two-layer bidirectional LSTM.
- A threshold parameter θ is used to transform probabilities into actual constraints:

$$i \in \bar{B} \;\text{ iff }\; P(\bar{B} \mid \mathbf{x}, i) > \theta$$

- We achieve a precision of over 99 % with a recall of far over 90 % for both classes $\bar{B}$ and $\bar{E}$.





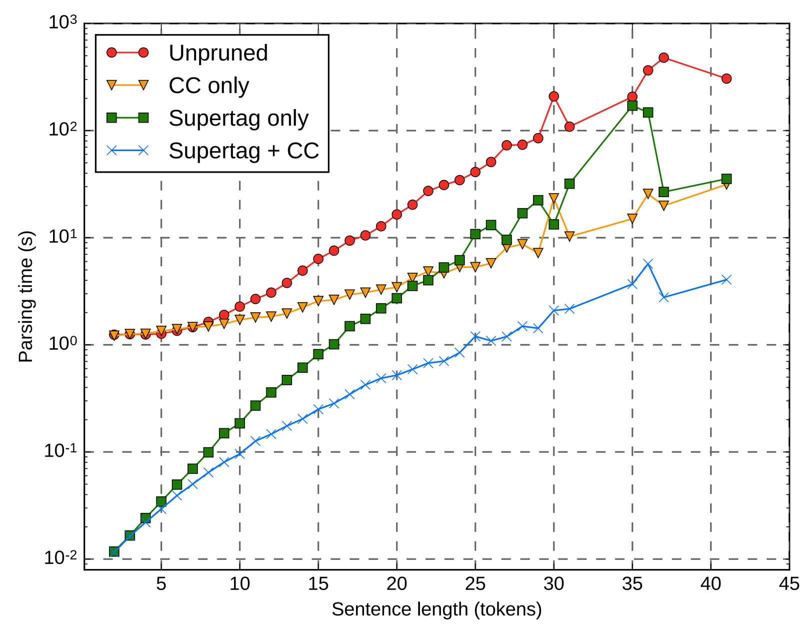
Evaluation: PCFG parsing

- We evaluate chart constraints for PCFG in combination with **coarse-to-fine parsing** (Charniak & Johnson, 2005; Teichmann et al., 2017).
- We parse Section 23 of the Penn Treebank, using POS tags as input.

Parser	f-score	time (ms)	speedup
Unpruned	71.0	2599	1.0x
CC ($\theta = 0.5$)	75.0	143	18.2x
CTF	67.6	194	13.4x
CTF + CC ($\theta = 0.5$)	72.4	37	**70.1x**

Evaluation: TAG parsing

- We evaluate chart constraints for TAG.
- Combine with neural **supertagger**, which predicts the k most likely elementary trees for each string position (cf. Bangalore & Joshi, 1999; Lewis et al., 2016).



- We convert the WSJ section of the Penn Treebank into a TAG corpus, removing multiple adjunction, and parse Section 23 of the converted corpus.

Parser	f-score	time (ms)	speedup
Unpruned	51.4	9483	1.0x
CC ($\theta = 0.95$)	53.6	2489	3.8x
supertag ($k = 3$)	78.5	132	72.0x
… + B/E (0.95)	79.2	87	108.9x
… + CC (0.95)	78.4	76	**124.3x**

Open-source implementation

For our experiments, we used the **Alto** parser (Gontrum et al., 2017) for Interpreted Regular Tree Grammars (IRTGs; Koller & Kuhlmann, 2011).
We are about to make our code available open-source as part of Alto, at `http://bitbucket.org/tclup/alto/`