Natural language syntax: parsing and complexity

Timothée Bernard and Pascal Amsili

Université Paris Cité, Université Sorbonne Nouvelle
timothee.bernard@u-paris.fr, pascal.amsili@ens.fr

Ljubljana, Slovenia – August 7-11, 2023
ESSLLI foundational course in Language and Computation

Overview of the course

- Day 1: Formal languages and syntactic complexity.
- Day 2: The complexity of natural language.
- Day 3: Historic algorithms for parsing.
- Day 4: Modern approaches to parsing.
- Day 5: Neural networks and error propagation.

Day 4

Recap from Day 3

- Top-down parsing: rewrite the axiom into the query.
- Bottom-up parsing: "unwrite" the query into the axiom.
- Shift-Reduce is a bottom-up transition system.
- Some (formal) languages have grammars that can be parsed deterministically.
- This is not possible with intrinsically ambiguous languages, such as natural languages.
- Chart-parsing methods (e.g. CYK, Earley) have $O(n^3)$ worst-case time complexity, even with ambiguous grammars

Today's contents

- Classifier-based transition parsing.
- Dependency parsing.
- Graph-based parsing.
- Classifier-based chart constituency parsing.
- CCG hybrid {classifier+grammar}-based parsing.

An ML component can replace a grammar

- **Grammar-based algorithm**: uses an explicit formal grammar.
- For NL, formal grammars are often replaced by a statistical component inferred from data by a machine-learning algorithm.
- Two popular families:
  - classifier-based parsing (with a classifier);
  - graph-based parsing (with a scorer).

Classifiers map states to actions

- **Classifier-based parser**: transition parser (e.g. Shift-Reduce) that relies on a classifier to select which action to apply).
- **Classifier**: function that maps each parsing state to an action.
- Sagae & Lavie 2005:
  - first occurrence of classifier-based constituency parsing;
  - variant of Shift-Reduce;
  - given a set of syntactic categories N, all Reduce($A \to BC$)·s and all Reduce($A \to B$)·s are available;
  - the input tokens are already tagged with POS tags from N;
  - constraints on unary Reduce·s $\to$ linear time complexity.

From symbolic to statistical vectors

- Classifiers typically work on vector representations of the states.
- Past: states encoded as sparse vectors of symbolic features. Examples (for SR):
  - identity of the word on top of the stack,
  - distance in the sentence between the token on top of the stack and the first token in the buffer.
- Present: states encoded as dense vectors by neural networks. ($\rightarrow$ Day 5)

Classifier-based parsers are trained using treebanks

- Training requires data.
- **Gold tree**: syntactic tree judged correct (e.g. obtained by expert annotation).
- **Treebank**: set of gold trees.
- Famously: the Penn Treebank (PTB; Marcus, Santorini & Marcinkiewicz 1993)
  - English text;
  - mainly from the news domain (WSJ);
  - $\sim$50.000 sentences, $\sim$1.000.000 tokens.

$\rightarrow$ Major impact on the statistical revolution in NLP.

Training a parser with teacher forcing is simple

- Given a treebank, a simple training algorithm:

Algorithm 1: Training of a transition parser (teacher forcing).

```
Function train(dataset)
    classifier = init_classif();                        // Initial (random) classifier.
    while ¬stop(classifier, dataset) do
        tree = dataset.get();
        a_1, a_2, ... a_n = oracle(tree);   // Seq. of actions that leads to the gold tree.
        state = init_state(tree);                       // Initial parsing state.
        for i := 1 to n do
            v = encode(state);                          // Vector representation.
            classifier.optim(v, a_i);
            state = state.apply(a_i);                   // The "teacher" action is forced.
    return classifier;
```

Teacher forcing is suboptimal

- Depending on the *encode* function, training with teacher forcing might be super fast (due to parallelism).
- But this kind of training has defects, e.g. *error propagation*.
- Slower but better: not only strengthen a gold derivation, but also weaken the predicted one (as is done within the *structured perceptron* paradigm; Collins 2002).
- More on how to fight error propagation in Day 5.

Classifier-based parsers can map any sequence to a tree

- Language recognised by a classifier-based parser: usually $\Sigma^\star$.
- But the mapping from input (a sequence of tokens) to output (a tree) can be arbitrarily complex.
- What happens with ungrammatical sentences?

  :/
- What happens with ambiguous sentences?

  The most “natural” structure is predicted.
- $\rightarrow$ Classifiers encode a grammar with *preferences*.

Quantifying the quality of constituent trees

- What matters: the quality of the analyses of grammatical sentences.

Evaluating constituency parsing

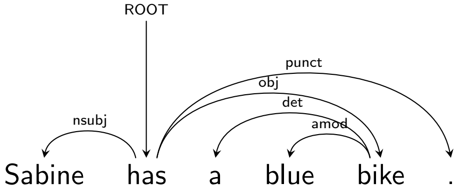
t_g: gold tree; t_p: predicted tree; both seen as sets of constituents

- **precision**: $p = \frac{|t_g \cap t_p|}{|t_p|}$
- **recall**: $r = \frac{|t_g \cap t_p|}{|t_g|}$
- **F1**: $\frac{2pr}{p+r}$

- SotA: $>$ 0.96 F1 on the PTB (e.g. Tian et al. 2020).

Syntactic dependency tree

- First occurrence of classifier-based parsing: in the context of **dependency parsing** (Kudo & Matsumoto 2002)
- **Dependency**:
  - *governor* $\overset{\textit{label}}{\rightarrow}$ *dependent*
  - syntactic relationship between a token (gov.) and another (dep.) that it legitimates (allows or requires)



A simple transition system for dependency parsing

- **Arc-Standard** (Nivre 2005, 2010):
  - one of the simplest transition systems for (projective) dependency parsing;
  - **shift**:

	(stack,	buffer,	dependencies)
from state	$(S$,	$w \mid B'$,	A)
to state	$(S \mid w$,	B',	A)

  - **left-arc**(l):

	(stack ,	buffer,	dependencies)
from state	$(S' \mid w_l \mid w_r$,	B,	A)
to state	$(S' \mid w_r$,	B,	$A + w_l \xleftarrow{l} w_r)$

  - **right-arc**(l):

	(stack ,	buffer,	dependencies)
from state	$(S' \mid w_l \mid w_r$,	B,	A)
to state	$(S' \mid w_l$,	B,	$A + w_l \xrightarrow{l} w_r)$

Illustration of Arc-Standard

[ROOT] [Sabine has a blue bike .]

Illustration of Arc-Standard

[ROOT Sabine] [has a blue bike .]

action: **shift**

Illustration of Arc-Standard

[ROOT Sabine has] [a blue bike .]

action: **shift**

Illustration of Arc-Standard

$$[\text{ROOT} \quad \text{Sabine} \quad \text{has}] \quad [\text{a} \quad \text{blue} \quad \text{bike} \quad .]$$

nsubj

action: **left-arc(nsubj)**

Illustration of Arc-Standard

$$[\text{ROOT} \quad \text{Sabine} \quad \text{has} \quad \text{a}] \quad [\text{blue} \quad \text{bike} \quad .]$$

nsubj

action: **shift**

Illustration of Arc-Standard

nsubj

[ROOT Sabine has a blue] [bike .]

action: **shift**

Illustration of Arc-Standard

nsubj

[ROOT Sabine has a blue bike] [.]

action: **shift**

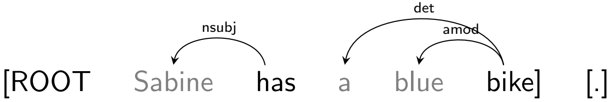
Illustration of Arc-Standard

nsubj amod

[ROOT Sabine has a blue bike] [.]

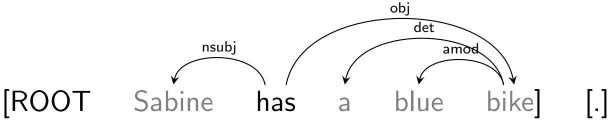
action: **left-arc(amod)**

Illustration of Arc-Standard



action: **left-arc(det)**

Illustration of Arc-Standard



action: **right-arc(obj)**

Illustration of Arc-Standard

obj det nsubj amod

[ROOT Sabine has a blue bike .] []

action: **shift**

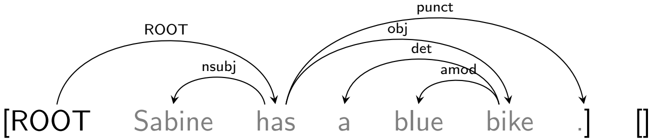
Illustration of Arc-Standard

punct
obj
det
nsubj amod

[ROOT Sabine has a blue bike .] []

action: **right-arc(punct)**

Illustration of Arc-Standard



action: **right-arc(ROOT)**

Quantifying the quality of dependency trees

Evaluating dependency parsing

t_g: gold tree; t_p: predicted tree; both seen as sets of dependency (Standard practice: ignore punctuation; Chen & Manning 2014.)

- **labelled attachment score** (LAS): $\frac{|t_g \cap t_p|}{|t_g|}$
- **unlabelled attachment score** (UAS): [similar but ignoring the dependency labels]

- SotA: $>$ 0.96 LAS on a conversion of the PTB (e.g. Mrini et al. 2020).
- Universal Dependencies project (UD; Nivre et al. 2016, de Marneffe et al. 2021): treebanks for $>$ 100 languages. `https://universaldependencies.org/`

Another paradigm: graph-based parsing

- **Graph-based parsing**:
  - other paradigm for dependencies (McDonald et al. 2005),
  - requires a *scorer*,
  - simple incarnation:
    1. score each candidate unlabelled dependency $w_i \rightarrow w_j$;
    2. compute $\text{argmax}_t \sum_{w_i \rightarrow w_j \in t} \mathit{score}(w_i \rightarrow w_j)$ with the maximum spanning tree (MST) algorithm;
    3. label each dependency with a classifier.
- Scorer trained to assign higher scores to gold trees.
- Effective implementation: Dozat & Manning 2017.
- (Many variations are possible.)

Scorers can be used in chart constituency parsing

- There are scorer-based versions of CYK.
- Gaddy, Stern & Klein 2018:
  1. score each candidate constituent $[X\ w_{i:j}]$;
  2. compute $\text{argmax}_t \sum_{[X\ w_{i:j}]\in t} \textit{score}([X\ w_{i:j}])$ with a variant of CYK:
     - $\forall i,\ T[i, i+1] = \max_X([X\ w_i])$
     - $\forall j > i+1,$
       $T[i,j] = \max_X([X\ w_{i:j}]) + \max_{i<k<j}(T[i,k] + T[k,j])$
- Scorer trained to assign higher scores to gold trees.

Combining grammars and classifiers is possible

- Is it possible to combine grammar-based and ML-based parsing?
  → Yes. (basic ex: constrain a classifier-based transition parser with a CFG)
- Lewis & Steedman (2014) use a two-step process for CCG parsing:
  1. classifier-based *supertagging*;
  2. grammar-based *decoding*.

$$\cfrac{\mathrm{Sabine}_1 \quad \cfrac{\mathrm{likes}_2}{\texttt{(S\textbackslash NP)/NP}} \quad \cfrac{\mathrm{books}_3}{\texttt{NP}}}{}$$

$$\dfrac{\dfrac{\text{Sabine}_1}{\mathtt{NP}} \qquad \dfrac{\dfrac{\text{likes}_2}{\mathtt{(S\backslash NP)/NP}} \quad \dfrac{\text{books}_3}{\mathtt{NP}}}{\mathrm{S}\backslash\mathrm{NP}}>}{\mathrm{S}}<$$

- Inspiration: Bangalore & Joshi (1999), who work with TAG.

Supertagging is a very informative classification task

- **Supertagging**:
  - for each token, predict a distribution of probability over all possible lexical categories (AKA **supertags**);
  - classification task learned from an annotated corpus (e.g. CCGbank; Hockenmaier & Steedman 2007);

  $\rightarrow \forall$ token w_i, $\forall$ supertag c, $P(C_i = c \mid w_{1:n})$.
- Harder than POS tagging, but with accurate supertagging, parsing is "almost done".

A* decoding for CCG (introduction)

- Lewis & Steedman (2014)'s decoding technique: bears some similarities with CYK and Earley, and also based on the *A* search algorithm*.
- Two data structures:
  - **chart**,
    - collection of **partial analyses** (tree covering a span $w_{i:j}$),
    - initially empty;
    - intuition: store for already done work;
  - **agenda**,
    - collection of partial analyses,
    - initialised as $\{ \frac{w_i}{c} \mid \forall$ token w_i, supertag $c\}$,
    - intuition: waiting queue for remaining work.

A* decoding for CCG (scores)

- **Score** of an analysis of $w_{i:j}$: product of two values defined based on the supertags used for $w_{i:j}$,
  - **internal score**,
$$\prod_{k=i}^{j} P(C_k = c_k \mid w_{1:n});$$
  - **external score**,
$$\prod_{k=1}^{i-1} \max_c P(C_k = c \mid w_{1:n}) \times \prod_{k=j+1}^{n} \max_c P(C_k = c \mid w_{1:n}).$$
- The score of an analysis is an upper bound of the score of any of its *extensions* (because the $\max\cdot$s are replaced by equal or lower probabilities).

A* decoding for CCG (overview)

1. Take the highest scoring partial analysis out of the agenda.
2. If this analysis covers the whole sentence, parsing is over: output this analysis.
3. Otherwise:
   - add this analysis to the chart,
   - add to the agenda all possible analyses obtained from it using any syntactic rule, combining it with other partial analyses found *in the chart* (and not in the agenda).
4. Go back to step 1.

$\rightarrow$ exact search (the output is the highest scoring analysis)

A* decoding for CCG (example)

- Simplified example on "Sabine likes books", forward/backward applications and three supertags only.
- Supertagging:
  - $P(c_1 = N \mid w_{1:4}) = 0.15$, $P(c_1 = NP \mid w_{1:4}) = 0.8$, $P(c_1 = (S\backslash NP)/NP \mid w_{1:4}) = 0.05$
  - $P(c_2 = N \mid w_{1:4}) = 0.05$, $P(c_2 = NP \mid w_{1:4}) = 0.05$, $P(c_2 = (S\backslash NP)/NP \mid w_{1:4}) = 0.9$
  - $P(c_3 = N \mid w_{1:4}) = 0.5$, $P(c_3 = NP \mid w_{1:4}) = 0.45$, $P(c_3 = (S\backslash NP)/NP \mid w_{1:4}) = 0.05$
- Initial chart: []
- Initial agenda: [$\dfrac{\text{likes}_2}{\mathtt{(S\backslash NP)/NP}}$ (0.9×0.4) , $\dfrac{\text{Sabine}_1}{\mathtt{NP}}$ (0.8×0.45) , $\dfrac{\text{books}_3}{\mathtt{N}}$ (0.5×0.72) , $\dfrac{\text{books}_3}{\mathtt{NP}}$ (0.45×0.72) , $\dfrac{\text{Sabine}_1}{\mathtt{N}}$ (0.15×0.45) , ...]

A* decoding for CCG (example)

- Chart: [$\dfrac{\text{likes}_2}{\mathtt{(S\backslash NP)/NP}}$]
- Agenda: [$\dfrac{\text{Sabine}_1}{\mathtt{NP}}$ (0.8×0.45) , $\dfrac{\text{books}_3}{\mathtt{N}}$ (0.5×0.72) , $\dfrac{\text{books}_3}{\mathtt{NP}}$ (0.45×0.72) , $\dfrac{\text{Sabine}_1}{\mathtt{N}}$ (0.15×0.45) , . . .]

A* decoding for CCG (example)

- Chart: [$\dfrac{\text{likes}_2}{\texttt{(S\textbackslash NP)/NP}}$, $\dfrac{\text{Sabine}_1}{\texttt{NP}}$]
- Agenda: [$\dfrac{\text{books}_3}{\texttt{N}}$ (0.5×0.72) , $\dfrac{\text{books}_3}{\texttt{NP}}$ (0.45×0.72) , $\dfrac{\text{Sabine}_1}{\texttt{N}}$ (0.15×0.45) , ...]

A* decoding for CCG (example)

- Chart: [$\dfrac{\text{likes}_2}{(S\backslash NP)/NP}$, $\dfrac{\text{Sabine}_1}{NP}$, $\dfrac{\text{books}_3}{N}$]
- Agenda: [$\dfrac{\text{books}_3}{NP}$ (0.45×0.72) , $\dfrac{\text{Sabine}_1}{N}$ (0.15×0.45) , ...]

A* decoding for CCG (example)

- Chart: [$\dfrac{\text{likes}_2}{\texttt{(S\textbackslash NP)/NP}}$, $\dfrac{\text{Sabine}_1}{\texttt{NP}}$, $\dfrac{\text{books}_3}{\texttt{N}}$, $\dfrac{\text{books}_3}{\texttt{NP}}$]
- Agenda: [$\dfrac{\dfrac{\text{likes}_2}{(S\backslash NP)/NP} \quad \dfrac{\text{books}_3}{NP}}{S\backslash NP}{>}$ (0.405×0.8) , $\dfrac{\text{Sabine}_1}{N}$ (0.15×0.45)

  , ...]

A* decoding for CCG (example)

- Chart: [$\dfrac{\text{likes}_2}{(S\backslash NP)/NP}$, $\dfrac{\text{Sabine}_1}{NP}$, $\dfrac{\text{books}_3}{N}$, $\dfrac{\text{books}_3}{NP}$, $\dfrac{\dfrac{\text{likes}_2}{(S\backslash NP)/NP} \quad \dfrac{\text{books}_3}{NP}}{S\backslash NP}{>}$]
- Agenda: [$\dfrac{\dfrac{\text{Sabine}_1}{NP} \quad \dfrac{\dfrac{\text{likes}_2}{(S\backslash NP)/NP} \quad \dfrac{\text{books}_3}{NP}}{S\backslash NP}{>}}{S}{<}$ (0.324×1) , $\dfrac{\text{Sabine}_1}{N}$ (0.15×0.45) , ...]

A* decoding for CCG (example)

- Chart: [$\dfrac{\text{likes}_2}{(\mathtt{S}\backslash\mathtt{NP})/\mathtt{NP}}$, $\dfrac{\text{Sabine}_1}{\mathtt{NP}}$, $\dfrac{\text{books}_3}{\mathtt{N}}$, $\dfrac{\text{books}_3}{\mathtt{NP}}$, $\dfrac{\dfrac{\text{likes}_2}{(\mathtt{S}\backslash\mathtt{NP})/\mathtt{NP}} \quad \dfrac{\text{books}_3}{\mathtt{NP}}}{\mathrm{S}\backslash\mathrm{NP}}>$]
- Agenda: [$\dfrac{\text{Sabine}_1}{\mathtt{N}}$ (0.15×0.45) , ...]
- Output: $\dfrac{\dfrac{\text{Sabine}_1}{\mathtt{NP}} \quad \dfrac{\dfrac{\text{likes}_2}{(\mathtt{S}\backslash\mathtt{NP})/\mathtt{NP}} \quad \dfrac{\text{books}_3}{\mathtt{NP}}}{\mathrm{S}\backslash\mathrm{NP}}>}{\mathrm{S}}<$

CCG parsing is useful

- CCG parsing for English works in practice.
- Intuition: There is a nice balance between the number of supertags ($\sim$400) and the number of syntactic rules ($\sim$20).
- CCG comes with a compositional syntax-semantics interface.
- $\rightarrow$ CCG powers state-of-the-art symbolic natural language inference (NLI) systems (e.g. Haruta, Mineshima & Bekki 2022).

Day 4: Summary

- An ML component (classifier/scorer) can replace a grammar.
- This is possible both in transition and chart parsing.
- Doing so requires data (usually treebanks).
- Dependency parsing is popular; either transition- or graph-based.
- For CCG and TAG: the supertagging+decoding paradigm combines a classifier and a grammar.
- Grammar-based parsing can be convenient for compositional semantics.

- Bangalore, Srinivas & Aravind K. Joshi. 1999. Supertagging: An Approach to Almost Parsing. *Computational Linguistics* 25(2). 237–265. https://aclanthology.org/J99-2004.
- Chen, Danqi & Christopher Manning. 2014. A Fast and Accurate Dependency Parser using Neural Networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 740–750. Doha, Qatar: Association for Computational Linguistics. http://www.aclweb.org/anthology/D14-1082.
- Collins, Michael. 2002. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, 1–8. Association for Computational Linguistics. https://doi.org/10.3115/1118693.1118694.
- Dozat, Timothy & Christopher D. Manning. 2017. Deep Biaffine Attention for Neural Dependency Parsing. In *5th International Conference on Learning Representations, ICLR 2017, Toulon,*

France, April 24-26, 2017, Conference Track Proceedings. https://openreview.net/forum?id=Hk95PK9le.

- Gaddy, David, Mitchell Stern & Dan Klein. 2018. What's Going On in Neural Constituency Parsers? An Analysis. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 999–1010. New Orleans, Louisiana: Association for Computational Linguistics. https://doi.org/10.18653/v1/N18-1091.
- Haruta, Izumi, Koji Mineshima & Daisuke Bekki. 2022. Implementing Natural Language Inference for comparatives. *Journal of Language Modelling* 10(1). Number: 1, 139–191. https://doi.org/10.15398/jlm.v10i1.294.
- Hockenmaier, Julia & Mark Steedman. 2007. CCGbank: A Corpus of CCG Derivations and Dependency Structures Extracted from the Penn Treebank. *Computational Linguistics* 33(3). 355–396. https://doi.org/10.1162/coli.2007.33.3.355.

- Kudo, Taku & Yuji Matsumoto. 2002. Japanese Dependency Analysis using Cascaded Chunking. In *COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*. `https://aclanthology.org/W02-2016` (5 July, 2023).
- Lewis, Mike & Mark Steedman. 2014. A* CCG Parsing with a Supertag-factored Model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 990–1000. Doha, Qatar: Association for Computational Linguistics. `https://doi.org/10.3115/v1/D14-1107`.
- Marcus, Mitchell P., Beatrice Santorini & Mary Ann Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics* 19(2). 313–330. `https://www.aclweb.org/anthology/J93-2004`.
- de Marneffe, Marie-Catherine, Christopher D. Manning, Joakim Nivre & Daniel Zeman. 2021. Universal Dependencies.

Computational Linguistics 47(2). 255–308.
https://doi.org/10.1162/coli_a_00402.

- McDonald, Ryan, Fernando Pereira, Kiril Ribarov & Jan Hajič. 2005. Non-Projective Dependency Parsing using Spanning Tree Algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, 523–530. Vancouver, British Columbia, Canada: Association for Computational Linguistics. https://aclanthology.org/H05-1066.
- Mrini, Khalil, Franck Dernoncourt, Quan Hung Tran, Trung Bui, Walter Chang & Ndapa Nakashole. 2020. Rethinking Self-Attention: Towards Interpretability in Neural Parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, 731–742. Online: Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.findings-emnlp.65.
- Nivre, Joakim. 2005. *Dependency Grammar and Dependency Parsing*. Tech. rep. MSI 05133. Växjö University, School of

Mathematics & Systems Engineering.
http://stp.lingfil.uu.se/~nivre/docs/05133.pdf.

- Nivre, Joakim. 2010. Dependency Parsing. *Language and Linguistics Compass* 4(3). 138–152. https://doi.org/10.1111/j.1749-818X.2010.00187.x. (29 November, 2021).
- Nivre, Joakim, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty & Daniel Zeman. 2016. Universal Dependencies v1: A Multilingual Treebank Collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, 1659–1666. Portorož, Slovenia: European Language Resources Association (ELRA). https://aclanthology.org/L16-1262.
- Sagae, Kenji & Alon Lavie. 2005. A Classifier-Based Parser with Linear Run-Time Complexity. In *Proceedings of the Ninth International Workshop on Parsing Technology*, 125–132.

Vancouver, British Columbia: Association for Computational Linguistics.
https://www.aclweb.org/anthology/W05-1513.

- Tian, Yuanhe, Yan Song, Fei Xia & Tong Zhang. 2020. Improving Constituency Parsing with Span Attention. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, 1691–1703. Online: Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.findings-emnlp.153.