A Minimal Span-Based Neural Constituency Parser

Mitchell Stern, Jacob Andreas, Dan Klein

CS 546 Paper Presentation

Boyin Zhang

Outline

1. Introduction
2. Background
3. Model
4. Algorithms
5. Training Details
6. Experiments
7. Conclusion

Intro: Overview

This paper:

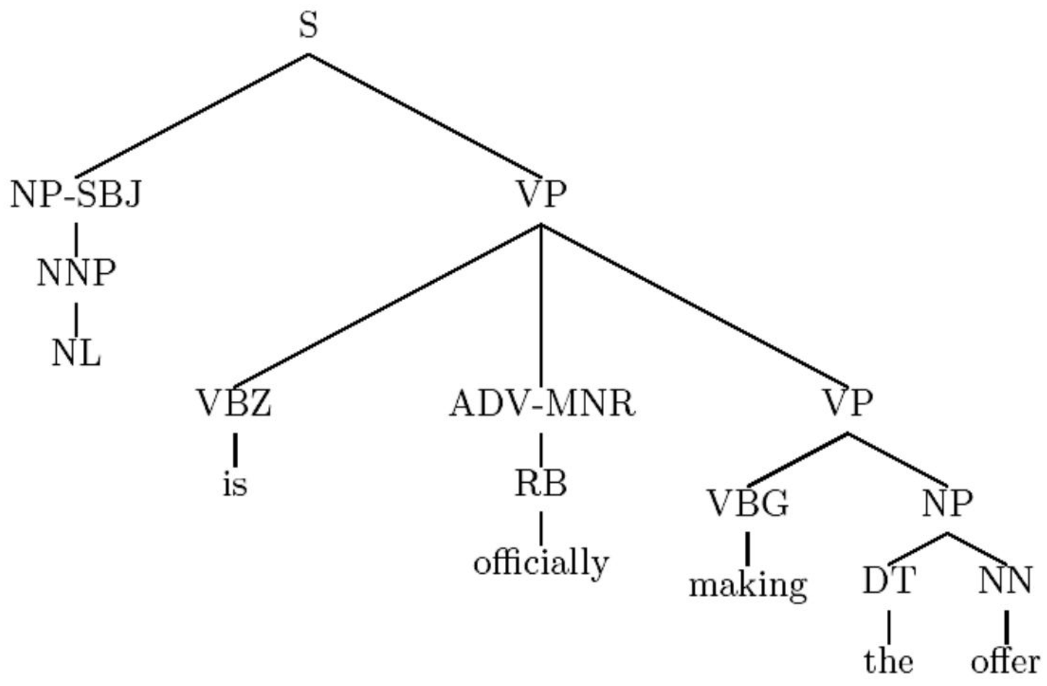
- constituency parsing
- a novel greedy top-down inference algorithm
- independent scoring for label and span

The goal is to preserve the basic algorithmic properties of span-oriented (rather than transition-oriented) parse representations, while exploring the extent to which neural representational machinery can replace the additional structure required by existing chart parsers.

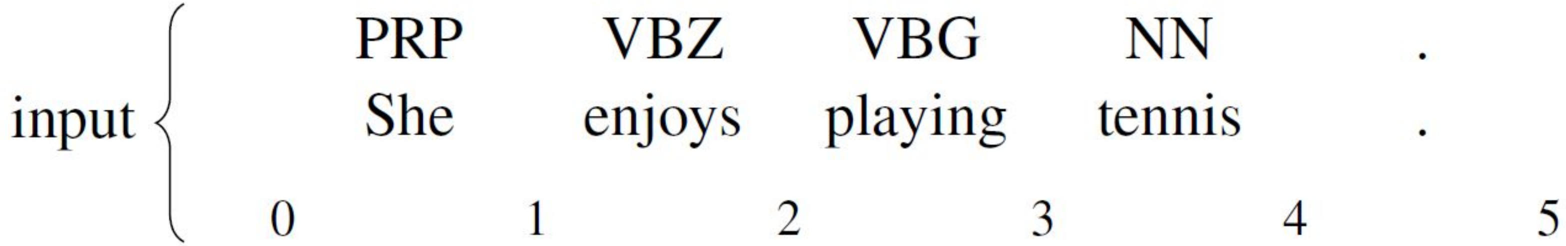
Intro: Penn Treebank

- The first publicly available syntactically annotated corpus
- Standard data set for English parsers
- Manually annotated with phrase-structure trees
- 48 preterminals (tags):
  - 36 POS tags, 12 other symbols (punctuation etc.)
- 14 nonterminals: standard inventory (S, NP, VP,...)
- Dataset for this paper

Intro: Constituency Parsing



Intro: Span and Label



span(0, 5) represent the full sentence, with label S.

Intro: Hinge Loss

In machine learning, the **hinge loss** is a loss function used for training classifiers. The hinge loss is used for "maximum-margin" classification, most notably for support vector machines (SVMs).[1]

Background: Transition Based Parser

- Do not admit fast dynamic programs and require careful feature engineering to support exact search-based inference (Thang et al., 2015)
- Require complex training procedures to benefit from anything other than greedy decoding (Wiseman and Rush, 2016)

Background: Chart Parser

- Require additional works, e.g, pre-specification of a complete context-free grammar for generating output structures and initial pruning of the output space
- Do not achieve results competitive with the best transition-based models.

Algorithm: Chart Parsing

The basic model, compatible with traditional chart-based dp algorithms.

$$T := \{(\ell_t, (i_t, j_t)) : t = 1, \ldots, |T|\},$$

$$s_{\text{tree}}(T) = \sum_{(\ell,(i,j)) \in T} \left[s_{\text{label}}(i, j, \ell) + s_{\text{span}}(i, j) \right].$$

Use modified CKY recursion to find the tree with highest score. O(n^3).

Model: Span Representation

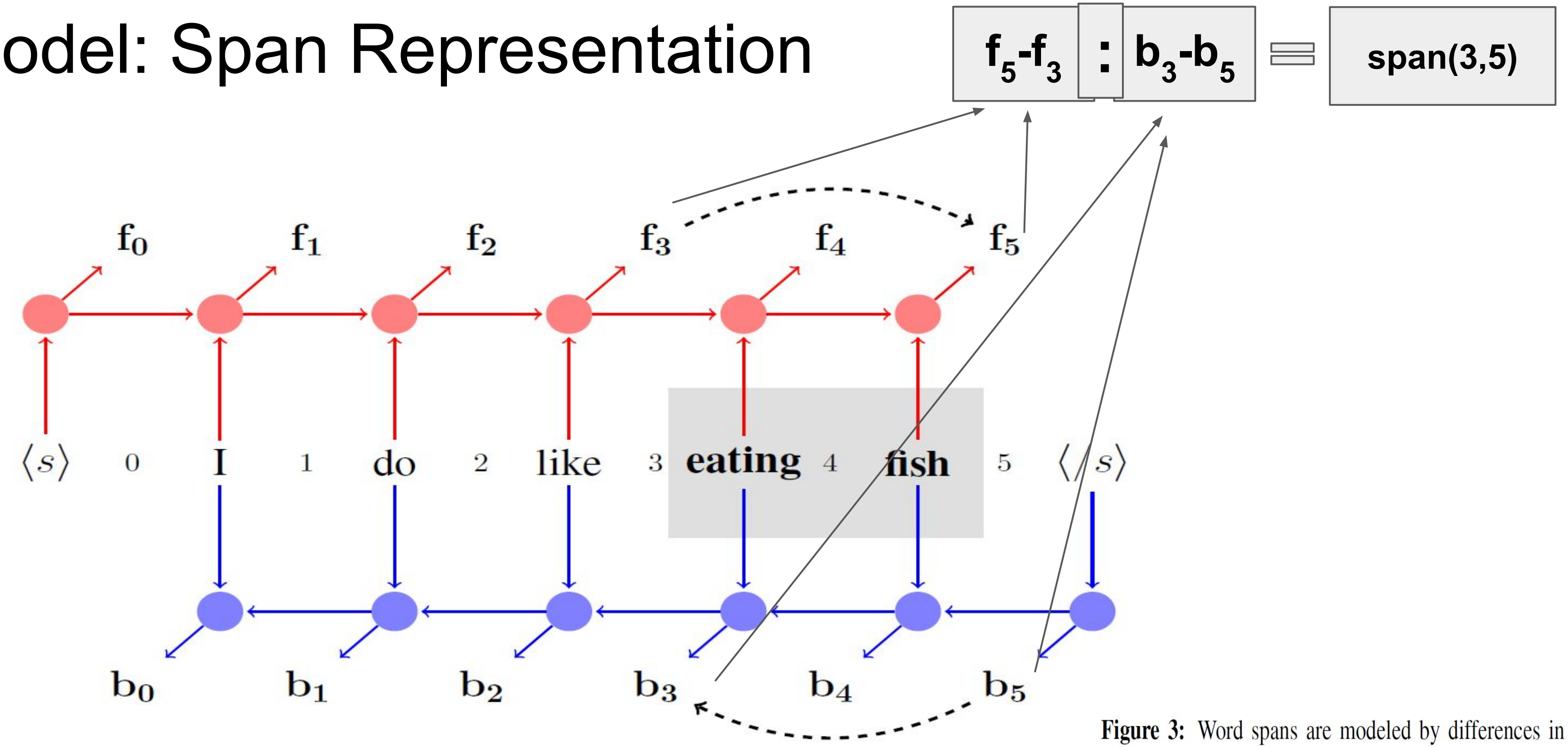


Figure 3: Word spans are modeled by differences in LSTM output. Here the span $_3$ *eating fish* $_5$ is represented by the vector differences ($f_5 - f_3$) and ($b_3 - b_5$). The forward difference corresponds to LSTM-Minus (Wang and Chang, 2016).

Model: Scoring Functions

$$s_{\text{labels}}(i, j) = \mathbf{V}_\ell g(\mathbf{W}_\ell \mathbf{s}_{ij} + \mathbf{b}_\ell),$$

$$s_{\text{span}}(i, j) = \mathbf{v}_s^\top g(\mathbf{W}_s \mathbf{s}_{ij} + \mathbf{b}_s),$$

$$s_{\text{label}}(i, j, \ell) = [s_{\text{labels}}(i, j)]_\ell,$$

Algorithm: Chart Parsing

- base case: $s_{\text{best}}(i, i+1) = \max_{\ell} [s_{\text{label}}(i, i+1, \ell)]$
- score of the split (i, k, j) as the sum of its subspan scores:

$$s_{\text{split}}(i, k, j) = s_{\text{span}}(i, k) + s_{\text{span}}(k, j).$$

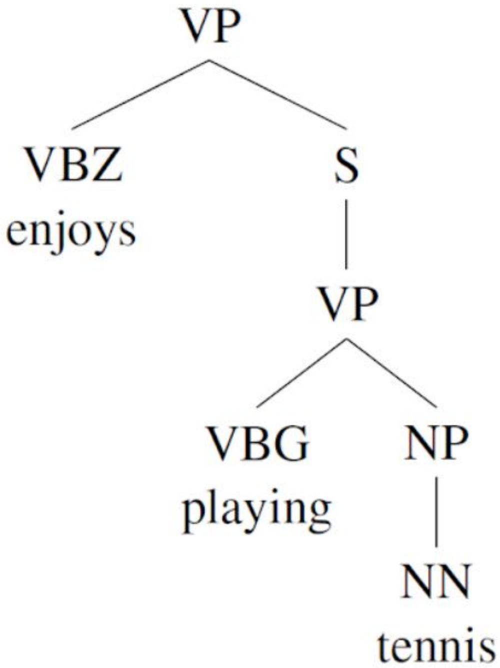
$$\tilde{s}_{\text{split}}(i, k, j) = s_{\text{split}}(i, k, j) + s_{\text{best}}(i, k) + s_{\text{best}}(k, j)$$

- joint label and split decision:

$$s_{\text{best}}(i, j) = \max_{\ell, k} [s_{\text{label}}(i, j, \ell) + \tilde{s}_{\text{split}}(i, k, j)]$$

$$s_{\text{best}}(i, j) = \max_{\ell} [s_{\text{label}}(i, j, \ell)] + \max_{k} [\tilde{s}_{\text{split}}(i, k, j)]$$

Algorithm: Chart Parsing



	PRP	VBZ	VBG	NN	.
	She	enjoys	playing	tennis	.
0	1	2	3	4	5

Finally, s_best(0, 5).

e.g. $s_{best}(1, 4)$: [(1, 2) (2, 4)]; [(1, 3) (3, 4)];

$$= \max[s_{label}(1,4)] + \max[(s_{best}(1, 2)+s_{best}(2, 4)+s_{span}(1, 2)+s_{span}(2, 4)),$$
$$(s_{best}(1, 3)+s_{best}(3, 4)+s_{span}(1, 3)+s_{span}(3, 4))]$$

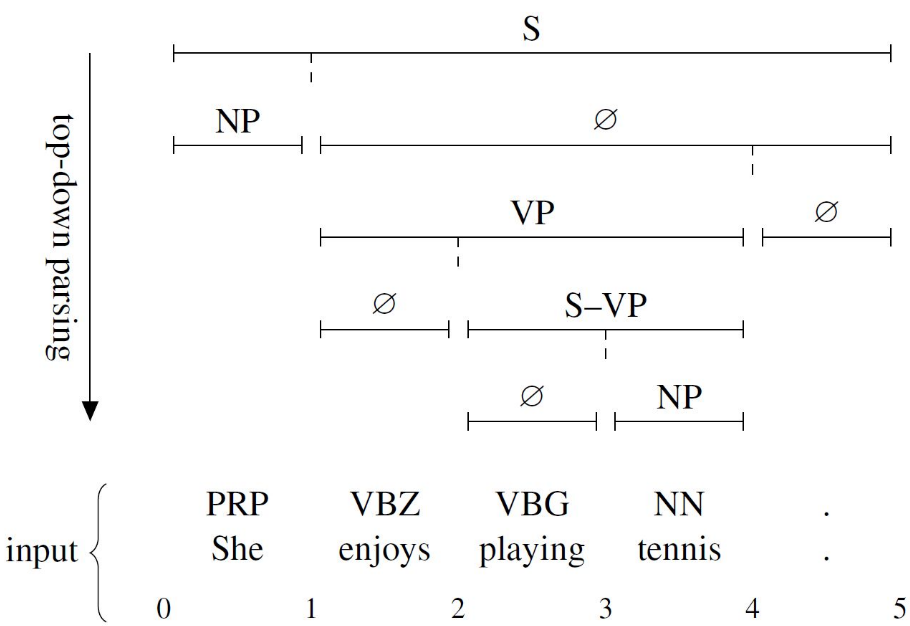
Algorithms: Top-Down Parsing

At a high level, given a span, we independently assign it a label and pick a split point, then repeat this process for the left and right subspans.

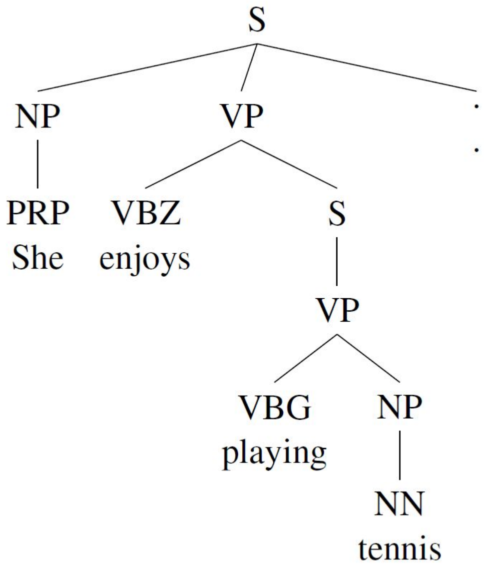
- base case: $\widehat{\ell} = \operatorname*{argmax}_{\ell} \left[s_{\text{label}}(i, i+1, \ell) \right]$
- label and split decision : $(\widehat{\ell}, \widehat{k}) = \operatorname*{argmax}_{\ell, k} \left[s_{\text{label}}(i, j, \ell) + s_{\text{split}}(i, k, j) \right]$

$$\widehat{\ell} = \operatorname*{argmax}_{\ell} \left[s_{\text{label}}(i, j, \ell) \right],$$
$$\widehat{k} = \operatorname*{argmax}_{k} \left[s_{\text{split}}(i, k, j) \right],$$

Algorithms: Top-Down Parsing



(a) Execution of the top-down parsing algorithm.



(b) Output parse tree.

Training: Loss Functions

For a span (i, j) occurring in the gold tree, let l* and k* represent the correct label and split point, and let $\hat{l}$ and $\widehat{k}$ be the predictions made by computing the maximizations

- Hinge loss for label: $$\max\left(0, 1 - s_{\text{label}}(i, j, \ell^*) + s_{\text{label}}(i, j, \widehat{\ell})\right)$$
- Hinge loss for split: $$\max\left(0, 1 - s_{\text{split}}(i, k^*, j) + s_{\text{split}}(i, \widehat{k}, j)\right)$$

Training: Alternatives

- Top-Middle-Bottom Label Scoring
- Left and Right Span Scoring
- Span Concatenation Scoring
- Deep Biaffine Span Scoring
- Structured Label Loss

Training: Details

- Penn Treebank for English experiments, French Treebank from the SPMRL 2014 shared task for French experiments.
- a two-layer bidirectional LSTM for our base span features. Dropout with a ratio selected from {0.2, 0.3, 0.4} is applied to all non-recurrent connections of the LSTM
- All parameters (including word and tag embeddings) are randomly initialized using Glorot initialization
- Adam optimizer with its default settings
- implemented in C++ using the DyNet neural network library (Neubig et al., 2017).

Evaluation Metric: F1 score

- The traditional F-measure or balanced F-score (**F_1 score**) is the harmonic mean of precision and recall

$$F_1 = 2 \cdot \frac{1}{\frac{1}{\mathrm{recall}} + \frac{1}{\mathrm{precision}}} = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}$$

Results

Final Parsing Results on Penn Treebank

Parser	LR	LP	F1
Durrett and Klein (2015)	–	–	91.1
Vinyals et al. (2015)	–	–	88.3
Dyer et al. (2016)	–	–	89.8
Cross and Huang (2016)	90.5	92.1	91.3
Liu and Zhang (2016)	91.3	92.1	91.7
Best Chart Parser	90.63	92.98	91.79
Best Top-Down Parser	90.35	93.23	91.77

Processing one sentence at a time on a c4.4xlarge Amazon EC2 instance:

- Chart parser: 20.3 sens/s
- Top-down: 75.5 sens/s

Conclusion

Span-Based Neural Constituency Parser

- bi-LSTM for span representation
- dynamic programming chart-based decoding
- a greedy novel top-down inference procedure
- NN methods works