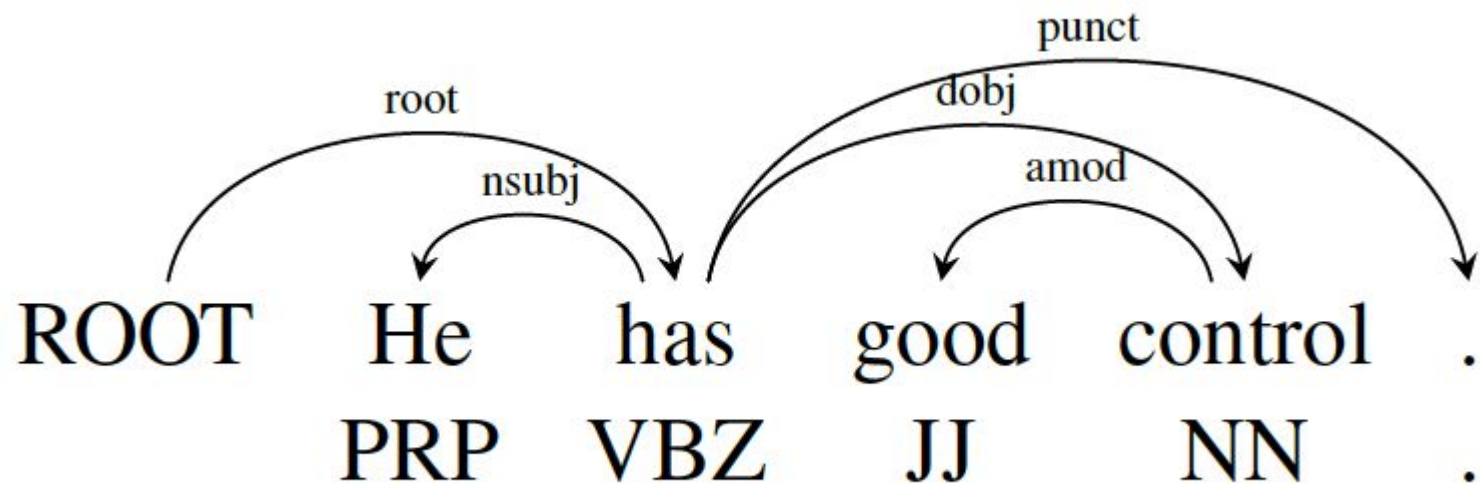


A Fast and Accurate Dependency Parser using Neural Networks

Danqi Chen, Christopher D. Manning. EMNLP 2014
Presented by Jessie Le (kle11), Spring 2020

Dependency Parser



Problem Statement

- Conventional feature-based discriminative dependency parsers have great success in dependency parsing.
- Limitations
 - Poorly estimated feature weights
 - Rely on a manually designed set of feature templates
 - Large time cost in the feature extraction step
- Solution
 - Use dense features in place of the sparse indicator features
 - Train a neural network classifier to make parsing decisions in a greedy, transition-based dependency parser

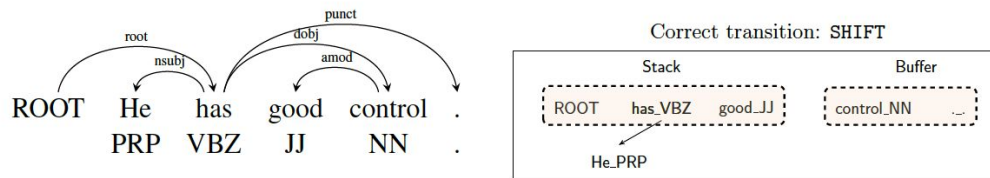
Greedy Transition-based Dependency Parser

- Goal: predict a correct transition from τ , based on one given configuration

Greedy Transition-based Dependency Parser

- Goal: predict a correct transition from τ , based on one given configuration
- Arc-standard system - one of the most popular transition system

- LEFT-ARC(l): adds an arc $s_1 \rightarrow s_2$ with label l and removes s_2 from the stack. Precondition: $|s| \geq 2$.
- RIGHT-ARC(l): adds an arc $s_2 \rightarrow s_1$ with label l and removes s_1 from the stack. Precondition: $|s| \geq 2$.
- SHIFT: moves b_1 from the buffer to the stack. Precondition: $|b| \geq 1$.



Transition	Stack	Buffer	A
	[ROOT]	[He has good control .]	\emptyset
SHIFT	[ROOT He]	[has good control .]	
SHIFT	[ROOT He has]	[good control .]	
LEFT-ARC (nsubj)	[ROOT has]	[good control .]	$A \cup \text{nsubj}(\text{has}, \text{He})$
SHIFT	[ROOT has good]	[control .]	
SHIFT	[ROOT has good control]	[.]	
LEFT-ARC (amod)	[ROOT has control]	[.]	$A \cup \text{amod}(\text{control}, \text{good})$
RIGHT-ARC (dobj)	[ROOT has]	[.]	$A \cup \text{dobj}(\text{has}, \text{control})$
...
RIGHT-ARC (root)	[ROOT]	[.]	$A \cup \text{root}(\text{ROOT}, \text{has})$

Figure 1: An example of transition-based dependency parsing. Above left: a desired dependency tree, above right: an intermediate configuration, bottom: a transition sequence of the arc-standard system.

Greedy Transition-based Dependency Parser

- Goal: predict a correct transition from τ , based on one given configuration
- Arc-standard system
- Conventional approaches: extract indicator features

Single-word features (9)

$s_1.w; s_1.t; s_1.wt; s_2.w; s_2.t;$
 $s_2.wt; b_1.w; b_1.t; b_1.wt$

Word-pair features (8)

$s_1.wt \circ s_2.wt; s_1.wt \circ s_2.w; s_1.wts_2.t;$
 $s_1.w \circ s_2.wt; s_1.t \circ s_2.wt; s_1.w \circ s_2.w$
 $s_1.t \circ s_2.t; s_1.t \circ b_1.t$

Three-word features (8)

$s_2.t \circ s_1.t \circ b_1.t; s_2.t \circ s_1.t \circ lc_1(s_1).t;$
 $s_2.t \circ s_1.t \circ rc_1(s_1).t; s_2.t \circ s_1.t \circ lc_1(s_2).t;$
 $s_2.t \circ s_1.t \circ rc_1(s_2).t; s_2.t \circ s_1.w \circ rc_1(s_2).t;$
 $s_2.t \circ s_1.w \circ lc_1(s_1).t; s_2.t \circ s_1.w \circ b_1.t$

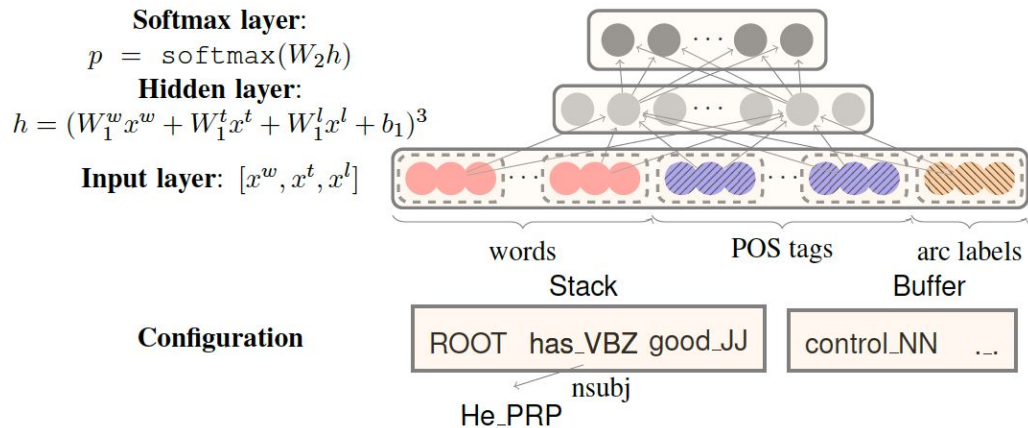
Table 1: The feature templates used for analysis. $lc_1(s_i)$ and $rc_1(s_i)$ denote the leftmost and rightmost children of s_i , w denotes word, t denotes POS tag.

Greedy Transition-based Dependency Parser

- Arc-standard system
- Goal: predict a correct transition from τ , based on one given configuration
- Conventional approaches: extract indicator features
- Problem of those indicator features
 - Sparsity
 - Incompleteness
 - Expensive feature computation

Model Architecture

- Input:
 - Word, POS tags, and arc labels embeddings
- Output:
 - Distribution of the transition

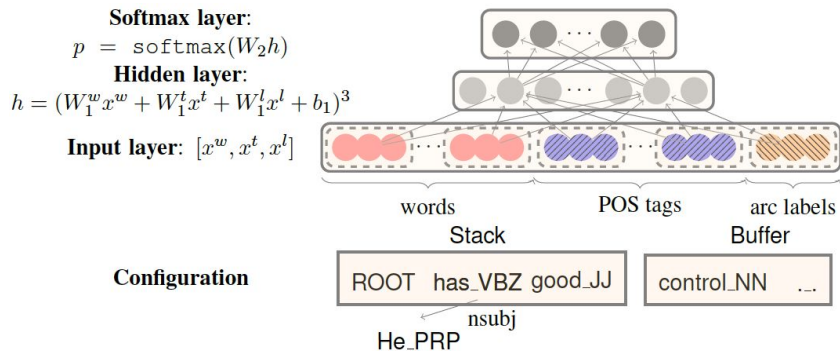


Model Architecture – Input Layer

- Represent each word as a d-dimensional vector $e_i^w \in \mathbb{R}^d$
- Word embedding matrix $E^w \in \mathbb{R}^{d \times N_w}$, where N_w is the dictionary size
- Map POS tags and arc labels to a d-dimensional vector space $e_i^t, e_j^l \in \mathbb{R}^d$
- POS embedding matrix is $E^t \in \mathbb{R}^{d \times N_t}$, where N_t is the number of distinct POS tags
- Label embedding matrix is $E^l \in \mathbb{R}^{d \times N_l}$, where N_l is the number of distinct arc labels

Model Architecture – Input Layer

- Chosen set depends on the stack or buffer positions S^w, S^t, S^l
- Example
 - $S^t = \{lc_1(s_2).t, s_2.t, rc_1(s_2).t, s_1.t\}$
 - PRP, VBZ, NULL, JJ
- $S^w = \{w_1, \dots, w_{n_w}\}$.
 $x^w = [e_{w_1}^w; e_{w_2}^w; \dots e_{w_{n_w}}^w]$ is added to the input layer
 where n_w is the number of chosen elements of the type of word



Model Architecture – Activation Function

- Cube activation function

- $h = (W_1^w x^w + W_1^t x^t + W_1^l x^l + b_1)^3$

- **Output** where $W_1^w \in \mathbb{R}^{d_h \times (d \cdot n_w)}$, $W_1^t \in \mathbb{R}^{d_h \times (d \cdot n_t)}$, $W_1^l \in \mathbb{R}^{d_h \times (d \cdot n_l)}$, and $b_1 \in \mathbb{R}^{d_h}$ is the bias.

- $p = \text{softmax}(W_2 h)$
, where $W_2 \in \mathbb{R}^{|\mathcal{T}| \times d_h}$

Softmax layer:

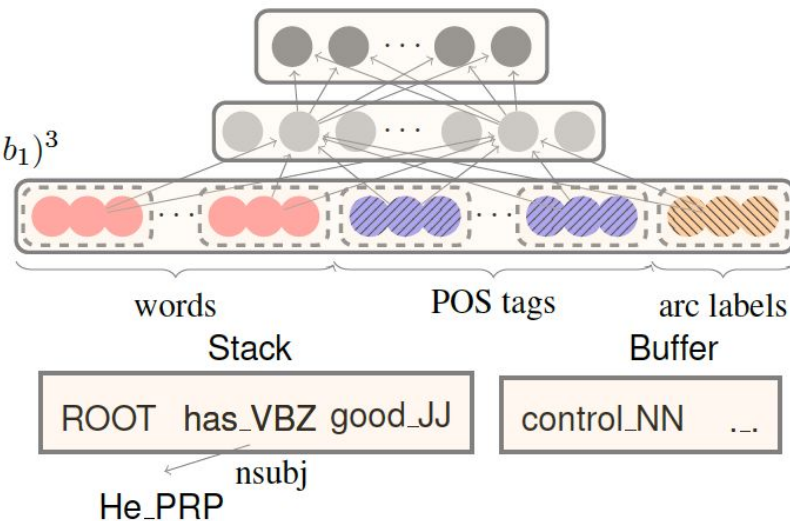
$$p = \text{softmax}(W_2 h)$$

Hidden layer:

$$h = (W_1^w x^w + W_1^t x^t + W_1^l x^l + b_1)^3$$

Input layer: $[x^w, x^t, x^l]$

$$g(w_1 x_1 + \dots + w_m x_m + b) = \sum_{i,j,k} (w_i w_j w_k) x_i x_j x_k + \sum_{i,j} b(w_i w_j) x_i x_j \dots$$



Experiments – Accuracy and Speed

- English Penn Treebank(PTB)
- Chinese Penn Treebank(CTB)
- MaltParser: a greedy transition-based dependency parser
 - stackproj(sp) -- arc-standard
 - nivreeager -- arc-eager
- MSTParser: a first-order graph-based parser

Parser	Dev		Test		Speed (sent/s)
	UAS	LAS	UAS	LAS	
standard	89.9	88.7	89.7	88.3	51
eager	90.3	89.2	89.9	88.6	63
Malt:sp	90.0	88.8	89.9	88.5	560
Malt:eager	90.1	88.9	90.1	88.7	535
MSTParser	92.1	90.8	92.0	90.5	12
Our parser	92.2	91.0	92.0	90.7	1013

Table 4: Accuracy and parsing speed on PTB + CoNLL dependencies.

Parser	Dev		Test		Speed (sent/s)
	UAS	LAS	UAS	LAS	
standard	90.2	87.8	89.4	87.3	26
eager	89.8	87.4	89.6	87.4	34
Malt:sp	89.8	87.2	89.3	86.9	469
Malt:eager	89.6	86.9	89.4	86.8	448
MSTParser	91.4	88.1	90.7	87.6	10
Our parser	92.0	89.7	91.8	89.6	654

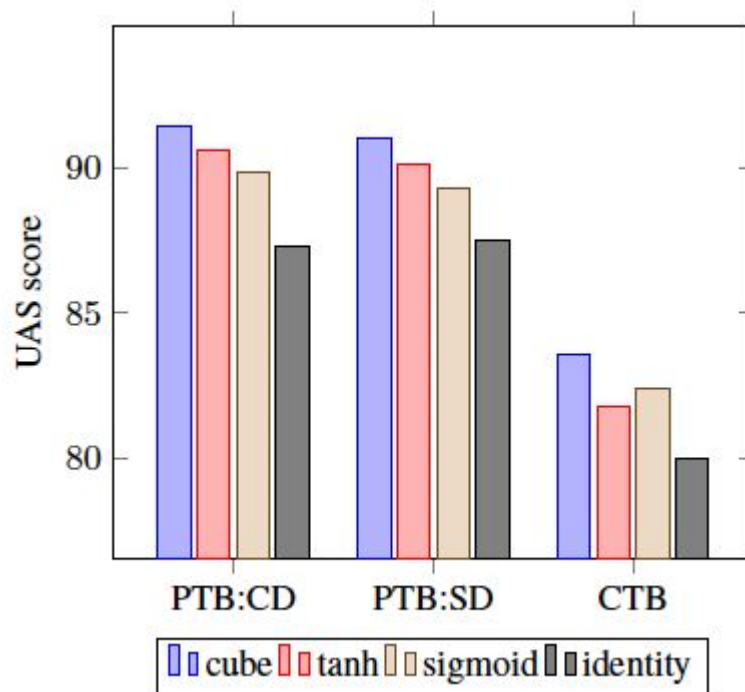
Table 5: Accuracy and parsing speed on PTB + Stanford dependencies.

Parser	Dev		Test		Speed (sent/s)
	UAS	LAS	UAS	LAS	
standard	82.4	80.9	82.7	81.2	72
eager	81.1	79.7	80.3	78.7	80
Malt:sp	82.4	80.5	82.4	80.6	420
Malt:eager	81.2	79.3	80.2	78.4	393
MSTParser	84.0	82.1	83.0	81.2	6
Our parser	84.0	82.4	83.9	82.4	936

Table 6: Accuracy and parsing speed on CTB.

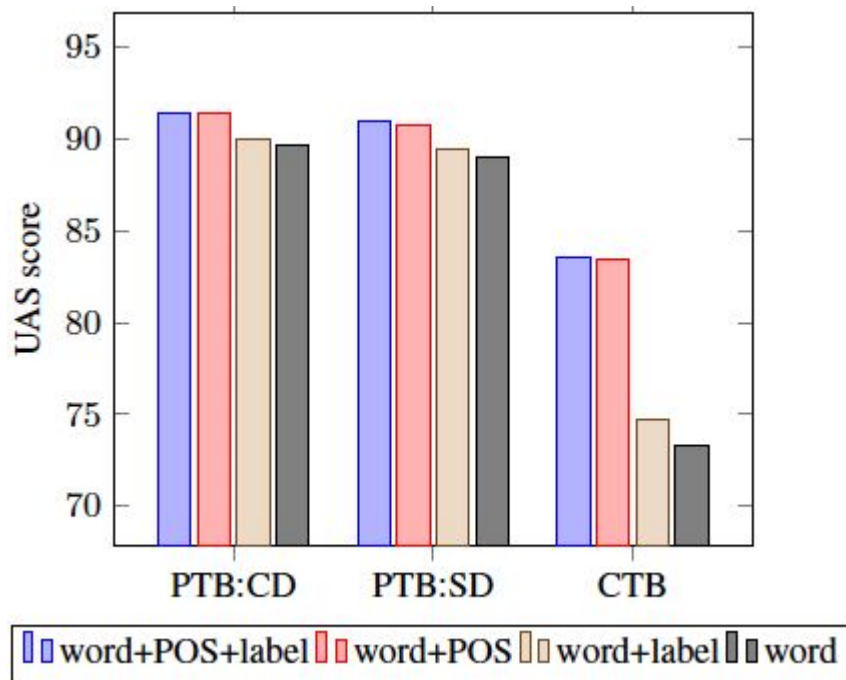
Experiments – Cube Activation Function

- **Cube** (x^3)
- $\tanh \left(\frac{e^x - e^{-x}}{e^x + e^{-x}} \right)$
- $\text{sigmoid} \left(\frac{1}{1 + e^{-x}} \right)$
- **Identity** (x)



Experiments – POS Tag & Arc Label Embeddings

- POS embeddings yield around
 - 1.7% improvement on PTB
 - 10% improvement on CTB
- Label embeddings yield around
 - 0.3% improvements on PTB
 - 1.4% improvement on CTB



Examination – POS Tag & Arc Labels Embeddings

- Encode the semantic regularities

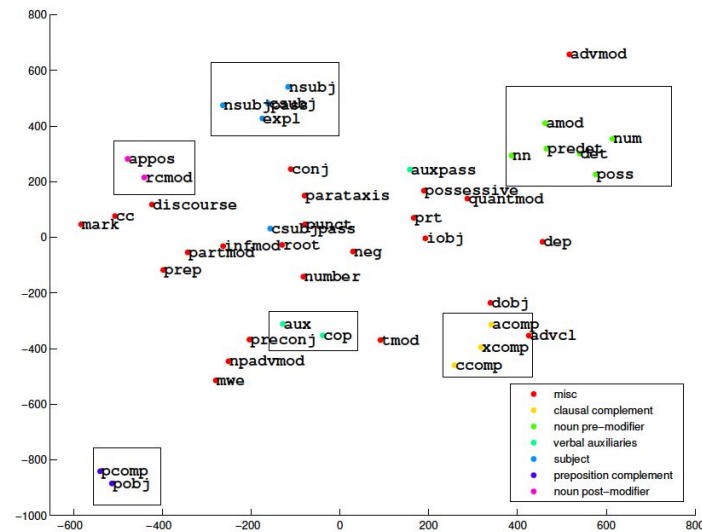
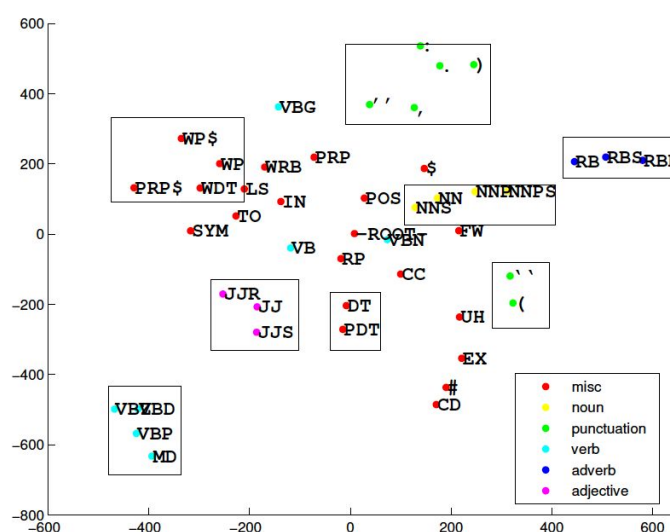


Figure 5: t-SNE visualization of POS and label embeddings.

Examination – Hidden Layer Weights

- POS tags weights in dependency parsing
- Identify useful information automatically
- Extract features not presented in the indicator feature system (>3)

- Feature 1: $s_1.t, s_2.t, lc(s_1).t$.
- Feature 2: $rc(s_1).t, s_1.t, b_1.t$.
- Feature 3: $s_1.t, s_1.w, lc(s_1).t, lc(s_1).l$.

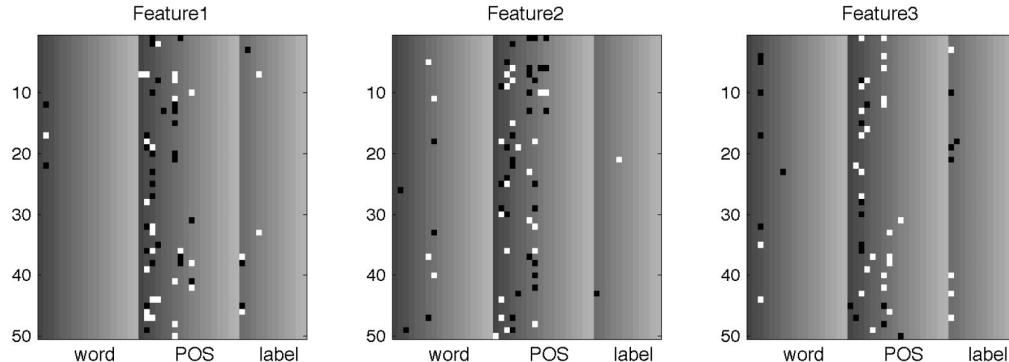


Figure 6: Three sampled features. In each feature, each row denotes a dimension of embeddings and each column denotes a chosen element, e.g., $s_1.t$ or $lc(s_1).w$, and the parameters are divided into 3 zones, corresponding to $W_1^w(k, :)$ (left), $W_1^t(k, :)$ (middle) and $W_1^l(k, :)$ (right). White and black dots denote the most positive weights and most negative weights respectively.

Conclusion

- Contribution
 - Introducing dense POS tag and arc label embedding into the input layer and show the usefulness within parsing task
 - Developing a NN architecture with good accuracy and speed
 - Cube activation function help capture high-order interaction feature
- Future work
 - Combine this classifier with search-based model
 - Theoretical studies on cube activation function
- Application

Citations

- [Carpuat, M. \(n.d.\). PDF.](#)
- Chen, D., & Manning, C. (2014). A Fast and Accurate Dependency Parser using Neural Networks. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). doi: 10.3115/v1/d14-1082
- Hale, R. (2019, February 15). Why we use dependency parsing. Retrieved March 23, 2020, from <https://www.megaputer.com/why-we-use-dependency-parsing/>
- Kohorst, L. (2019, December 13). Constituency vs. Dependency Parsing. Retrieved March 23, 2020, from <https://medium.com/@lucaskohorst/constituency-vs-dependency-parsing-8601986e5a52>
- Socher, R., Karpathy, A., Le, Q. V., Manning, C. D., & Ng, A. Y. (2014). Grounded Compositional Semantics for Finding and Describing Images with Sentences. Transactions of the Association for Computational Linguistics, 2, 207–218. doi: 10.1162/tacl_a_00177



Neural CRF Parsing

Authors:

Greg Durrett
Dan Klein

Presenter:

Rishabh Vaish

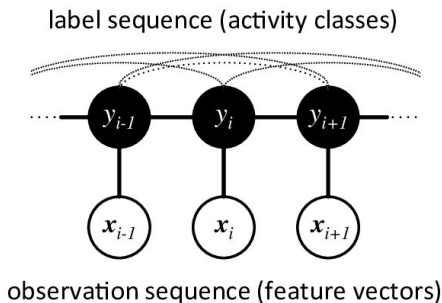
Outline

- What is CRF
- Overview
- Prior work
- Model
- Features
- Learning
- Inference
- Results
- Conclusion



What is CRF?

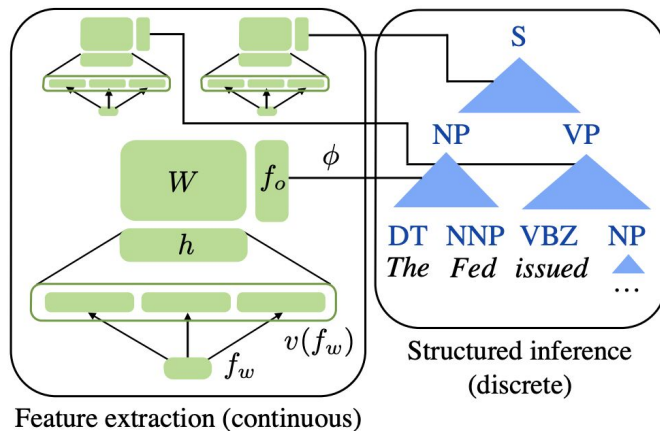
- Conditional Random Field
 - A class of statistical modeling method used for structured prediction
 - In NLP, it's major use case is for labeling of sequential data
 - Used to get a posterior probability of a label sequence conditioned on the input observation sequence
 - $P(\text{Label sequence } Y \mid \text{Observation sequence } X)$
 - Probability of change in label may depend on past and future observations
- Baseline CRF model was created by Hall et al. (2014)





Overview

- This paper presents a parsing model which combines the exact dynamic programming of CRF parsing with nonlinear featurization of feedforward neural networks.
- Model - Decomposes over anchored rules, and it scores each to these with a potential function - Nonlinear functions of word embeddings combined with linear functions of sparse indicator features like standard CRF





CFG (Rule)

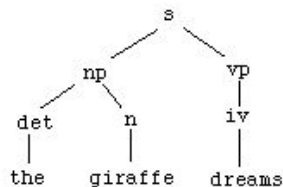
- A context-free grammar (CFG) is a list of rules that define the set of all well-formed sentences in a language.
- Example - The rule $s \rightarrow np\ vp$ means that "a sentence is defined as a noun phrase followed by a verb phrase."

```
s  → np vp
np  → det n
vp  → tv np
     → iv

det → the
    → a
    → an

n  → giraffe
   → apple

iv → dreams
tv → eats
   → dreams
```



- np - noun phrase
- vp - verb phrase
- s - sentence
- det - determiner (article)
- n - noun
- tv - transitive verb (takes an object)
- iv - intransitive verb
- prep - preposition
- pp - prepositional phrase
- adj - adjective

E.g., a parse of the sentence "the giraffe dreams" is:

$s \Rightarrow np\ vp \Rightarrow det\ n\ vp \Rightarrow the\ n\ vp \Rightarrow the\ giraffe\ vp \Rightarrow the\ giraffe\ iv \Rightarrow the\ giraffe\ dreams$



Anchored Rule

- It is the fundamental unit that the model considers. A tuple **(r,s)**
 - **r** - an indicator of rules identity
 - **s** - (i, j, k) indicates the span (i, k) and split point j of the rule
- A tree T is simply a collection of anchored rules subject to the constraint that those rules form a tree.
- All the parsing models are CRF that decompose over anchored rule productions and place a probability distribution over trees conditioned on a sentence as follows:

$$P(T|\mathbf{w}) \propto \exp \left(\sum_{(r,s) \in T} \phi(\mathbf{w}, r, s) \right)$$

where ϕ is the scoring function

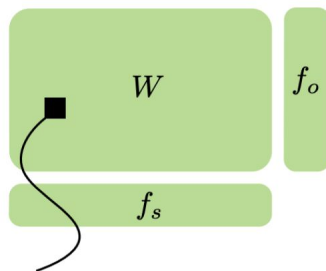


Scoring Anchored Rule (Sparse)

- ϕ considers the input sentence and the anchored rule in question.
- It can be a neural net, a linear function of surface features, or a combination of the two
- Baseline sparse scoring function -
 - $f_o(r) \in \{0, 1\}^{n_o}$ is a sparse vector of features expressing properties of r (such as the rule's identity or its parent label)
 - $f_s(w, s) \in \{0, 1\}^{n_s}$ is a sparse vector of surface features associated with the words in the sentence and the anchoring
 - W is an $n_s \times n_o$ matrix of weights

$$\phi_{\text{sparse}}(\mathbf{w}, r, s; W) = f_s(\mathbf{w}, s)^\top W f_o(r)$$

$$\text{a) } \phi = f_s^\top W f_o$$



$$W_{ij} = \text{weight}([f_{s,i} \wedge f_{o,j}]))$$



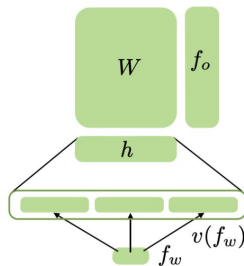
Scoring Anchored Rule (Neural)

- Neural scoring function -
 - $f_w(\mathbf{w}, s) \in \mathbb{N}^{n_w}$ to be a function that produces a fixed-length sequence of word indicators based on the input sentence and the anchoring
 - An embedding function $v: \mathbb{N} \rightarrow \mathbb{R}^{n_e}$, the dense representations of the words are subsequently concatenated to form a vector we denote by $v(f_w)$.
 - A matrix $H \in \mathbb{R}^{n_h \times (n_w n_e)}$ of real valued parameters
 - An element wise nonlinearity $g(\cdot)$, authors use rectified linear units $g(x) = \max(x, 0)$

$$h(\mathbf{w}, s; H) = g(Hv(f_w(\mathbf{w}, s)))$$

$$\phi_{\text{neural}}(\mathbf{w}, r, s; H, W) = h(\mathbf{w}, s; H)^\top W f_o(r)$$

$$\text{b) } \phi = g(Hv(f_w))^\top W f_o$$

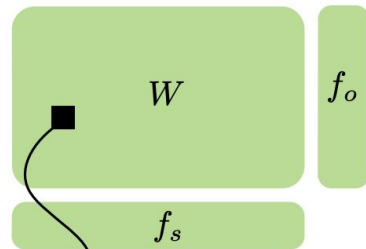




Scoring Anchored Rule (Combined)

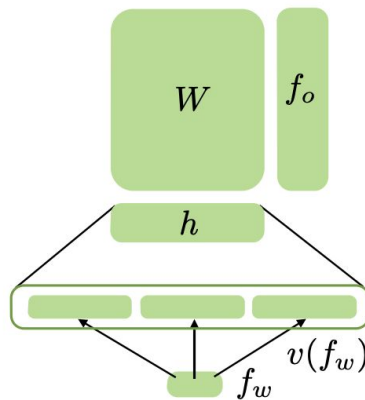
$$\phi(\mathbf{w}, r, s; W_1, H, W_2) = \phi_{\text{sparse}}(\mathbf{w}, r, s; W_1) \\ + \phi_{\text{neural}}(\mathbf{w}, r, s; H, W_2)$$

a) $\phi = f_s^\top W f_o$



$$W_{ij} = \text{weight}([f_{s,i} \wedge f_{o,j}])$$

b) $\phi = g(Hv(f_w))^\top W f_o$





Features

- Baseline Features (Sparse) f_s
 - Preterminal Layer
 - Prefixes and suffixes up to length 5 of the current word and neighboring words, as well as the words' identities
 - Nonterminal Productions
 - Fire indicators on the words before and after the start, end, and split point of the anchored rule
 - Span Properties - Span length + Span shape (an indicator of where capitalized words, numbers, and punctuation occur in the span)
- Neural Features
 - f_w - The words surrounding the beginning and end of a span and the split point (2 words in each direction)
 - v - Use pre-trained word vectors from Bansal et al. (do not update these vectors during training)



Learning (Gradient Descent)

- Maximize the conditional log-likelihood of our D training trees T^*

$$\mathcal{L}(H, W) = \sum_{i=1}^D \log P(T_i^* | \mathbf{w}_i; H, W)$$

- The gradient of W takes the standard form of log-linear models.

$$\frac{\partial \mathcal{L}}{\partial W} = \left(\sum_{(r,s) \in T^*} h(\mathbf{w}, s; H) f_o(r)^\top \right) - \left(\sum_T P(T | \mathbf{w}; H, W) \sum_{(r,s) \in T} h(\mathbf{w}, s; H) f_o(r)^\top \right)$$

Since h is the output of the neural network, we can first compute the following then apply the chain rule to get gradient for H

$$\frac{\partial \mathcal{L}}{\partial h} = \left(\sum_{(r,s) \in T^*} W f_o(r) \right) - \left(\sum_T P(T | \mathbf{w}; H, W) \sum_{(r,s) \in T} W f_o(r) \right)$$



Learning parameters

- Momentum term $\rho = 0.95$ (Zeiler (2012))
- A minibatch size of 200 trees
 - For each treebank, train for either 10 passes through the treebank or 1000 mini-batches, whichever is shorter
- Initialize the output weight matrix W to zero
- To break symmetry, the lower level neural network parameters H were initialized with each entry being independently sampled from a Gaussian with mean 0 and variance 0.01.
 - Gaussian performed better than uniform initialization, but the variance was not important.



Inference

- Speed up inference by using a coarse pruning pass (Hall et al. (2014))-
 - Prune according to an X-bar grammar with head outward binarization, ruling out any constituent whose max marginal probability is less than e^{-9}
 - Reduces the number of spans and split
- Note that the same word will appear in the same position in a large number of span/split point combinations, and cache the contribution to the hidden layer caused by that word (Chen and Manning, 2014)



Results - System

- Penn Treebank

	Sparse	Neural	V	Word Reps	$F_1 \text{ len} \leq 40$	$F_1 \text{ all}$
Hall et al. (2014), $V = 1$					90.5	
a	✓		0		89.89	89.22
b	✓		1		90.82	90.13
c	✓		1	Brown	90.80	90.17
d		✓	0	Bansal	90.97	90.44
e		✓	0	Collobert	90.25	89.63
f		✓	0	PTB	89.34	88.99
g	✓	✓	0	Bansal	92.04	91.34
h	✓	✓	0	PTB	91.39	90.91

We compare variants of our system along two axes: whether they use standard linear sparse features, nonlinear dense features from the neural net, or both, and whether any word representations (vectors or clusters) are used.

- Sparse (a and b) vs **Neural (d)**
- Wikipedia-trained word embeddings (e) vs **Vectors (d)**
- Continuous word representations (f) vs **Vectors (d)**
- (f) + sparse (h) vs **Vectors (d)**



Results - Design

- Penn Treebank

		$F_1 \text{ len} \leq 40$	Δ
Neural CRF		90.97	—
Nonlinearity	ReLU	90.97	—
	Tanh	90.74	−0.23
	Cube	89.94	−1.03
Depth	0 HL	90.54	−0.43
	1 HL	90.97	—
	2 HL	90.58	−0.39
Embed output		88.81	−2.16

To analyze the particular design choices we made for this system by examining the performance of several variants of the neural net architecture used -

- Choice of nonlinearity - Rectified linear units perform better than tanh or cubic units
- Depth - a network with one hidden layer performs best



Results

- Penn Treebank

	F ₁ all
Single model, PTB only	
Hall et al. (2014)	89.2
Berkeley	90.1
Carreras et al. (2008)	91.1
Shindo et al. (2012) single	91.1
Single model, PTB + vectors/clusters	
Zhu et al. (2013)	91.3
This work*	91.1
Extended conditions	
Charniak and Johnson (2005)	91.5
Socher et al. (2013)	90.4
Vinyals et al. (2014) single	90.5
Vinyals et al. (2014) ensemble	91.6
Shindo et al. (2012) ensemble	92.4

When sparse indicators are used in addition, the resulting model gets 91.1 F1 on section 23 of the Penn Treebank, outperforming the parser of Socher et al. (2013) as well as the Berkeley Parser (Petrov and Klein, 2007) and matching the discriminative parser of Carreras et al. (2008), and the single TSG parser of Shindo et al. (2012).



Results

- SPMRL (Nine other languages)

	Arabic	Basque	French	German	Hebrew	Hungarian	Korean	Polish	Swedish	Avg
Dev, all lengths										
Hall et al. (2014)	78.89	83.74	79.40	83.28	88.06	87.44	81.85	91.10	75.95	83.30
This work*	80.68	84.37	80.65	85.25	89.37	89.46	82.35	92.10	77.93	84.68
Test, all lengths										
Berkeley	79.19	70.50	80.38	78.30	86.96	81.62	71.42	79.23	79.18	78.53
Berkeley-Tags	78.66	74.74	79.76	78.28	85.42	85.22	78.56	86.75	80.64	80.89
Crabbé and Seddah (2014)	77.66	85.35	79.68	77.15	86.19	87.51	79.35	91.60	82.72	83.02
Hall et al. (2014)	78.75	83.39	79.70	78.43	87.18	88.25	80.18	90.66	82.00	83.17
This work*	80.24	85.41	81.25	80.95	88.61	90.66	82.23	92.97	83.45	85.08
Reranked ensemble										
2014 Best	81.32	88.24	82.53	81.66	89.80	91.72	83.81	90.50	85.50	86.12

Improvements on the performance of the parser from Hall et al. (2014) as well as the top single parser from the shared task (Crabbe and Seddah, 2014), with robust improvements on all languages



Conclusion

- Compared to Conventional CRF
 - Scores are non-linear potentials analogous to linear potential in conventional CRFs
 - Computations are factored along the same substructure as in conventional CRFs
- Compared to Prior neural network models
 - Removed the problem of structural prediction by making sequential decisions or by reranking
 - Authors framework allows exact inference via CKY because the potentials are still local to anchored rules.
- Shows significant improvement for English and nine other languages

Thank You

What's Going On in Neural Constituency Parsers?

An Analysis

David Gaddy, Mitchell Stern, and Dan Klein
University of California, Berkeley (NAACL 2018)

Adam J. Stewart

Department of Computer Science
University of Illinois at Urbana-Champaign

March 25, 2020

Summary

- 1 Motivation
- 2 Background
- 3 Model & Data
- 4 Experiments & Results
 - Output Correlations
 - Lexical Representation
 - Context in the Sentence LSTM
- 5 Discussion

Motivation

Natural Language Processing Has Evolved!

Classic NLP (explicit)

- Grammars
- Rich lexicons
- Handcrafted lexical features

Grammar	Lexicon
$S \rightarrow NP VP$	$Det \rightarrow that \mid this \mid the \mid a$
$S \rightarrow Aux NP VP$	$Noun \rightarrow book \mid flight \mid meal \mid money$
$S \rightarrow VP$	$Verb \rightarrow book \mid include \mid prefer$
$NP \rightarrow Pronoun$	$Pronoun \rightarrow I \mid she \mid me$
$NP \rightarrow Proper-Noun$	$Proper-Noun \rightarrow Houston \mid NWA$
$NP \rightarrow Det Nominal$	$Aux \rightarrow does$
$Nominal \rightarrow Noun$	$Preposition \rightarrow from \mid to \mid on \mid near \mid through$
$Nominal \rightarrow Nominal Noun$	
$Nominal \rightarrow Nominal PP$	
$VP \rightarrow Verb$	
$VP \rightarrow Verb NP$	
$VP \rightarrow Verb NP PP$	
$VP \rightarrow Verb PP$	
$VP \rightarrow VP PP$	
$PP \rightarrow Preposition NP$	

Modern NLP (implicit)

- Machine Learning
- RNNs, LSTMs, and GRUs
- Transformers

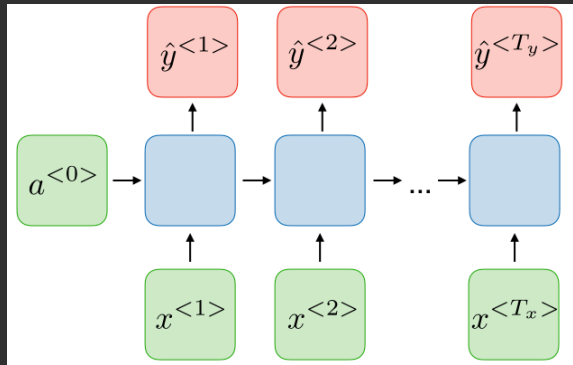
Natural Language Processing Has Evolved!

Classic NLP (explicit)

- Grammars
- Rich lexicons
- Handcrafted lexical features

Modern NLP (implicit)

- Machine Learning
- RNNs, LSTMs, and GRUs
- Transformers

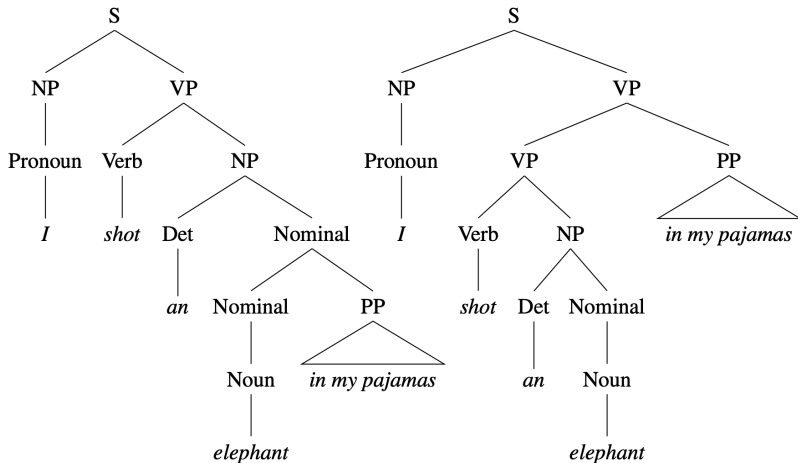


Research Questions

- To what extent is information provided directly by the model structure in classical systems still being captured by neural methods?
- How do RNNs compensate for the removal of the structures used in past models?
- What information do RNNs capture, and what is important for strong performance?

Background

What is Constituency Parsing?



Mathematical Framework

Scoring function $s(i, j, l)$ that assigns a real-valued score to every label l for each span (i, j)

Score of tree T is given by:

$$s(T) = \sum_{(i,j,l) \in T} s(i, j, l)$$

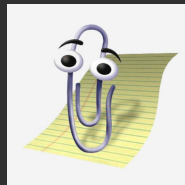
Parsing problem is to find optimal tree:

$$\hat{T} = \arg \max_T s(T)$$

Model & Data

Word Representation

- 1 Start with static word embedding
 - Embedding for every word in training set
 - Unknown words in test set mapped to <UNK>
- 2 Add character-level representation
 - Bidirectional character LSTM
 - Concatenate forward and backward outputs
- 3 Concatenate word embedding and character representation
 - $\mathbf{r} = [\mathbf{w}, \mathbf{c}_f, \mathbf{c}_b]$



Span Representation

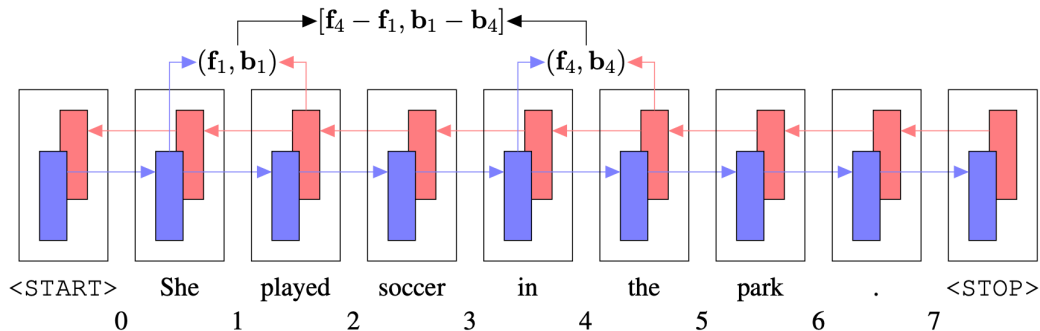


Figure 1: Span representations are computed by running a bidirectional LSTM over the input sentence and taking differences of the output vectors at the two endpoints. Here we illustrate the process for the span (1, 4) corresponding to “played soccer in” in the example sentence.

Label Scoring

Span representation given by:

$$\mathbf{r}_{ij} = [\mathbf{f}_j - \mathbf{f}_i, \mathbf{b}_i - \mathbf{b}_j]$$

Scoring function is single-layer feedforward network with output dimensionality equal to the number of possible labels

Score of specific label l is corresponding component of output vector:

$$s(i, j, l) = [\mathbf{W}_2 g(\mathbf{W}_1 \mathbf{r}_{ij} + \mathbf{z}_1) + \mathbf{z}_2]_l$$

where g is an elementwise ReLU nonlinearity

Inference

Inference: choosing the optimal tree

Problem: efficient CKY parsers only work for binary trees, but model uses n -ary trees

Solution: introduce auxiliary empty label \emptyset with $s(i, j, \emptyset) = 0$ to transform n -ary trees into binary trees

Book	the	flight	through	Houston
S, VP, Verb Nominal, Noun [0,1]	[0,2]	S,VP,X2 [0,3]	[0,4]	S,VP,X2 [0,5]
	Det [1,2]	NP [1,3]	[1,4]	NP [1,5]
		Nominal, Noun [2,3]	[2,4]	Nominal [2,5]
			Prep [3,4]	PP [3,5]
				NP, Proper- Noun [4,5]

Dataset

Penn Treebank (PTB) dataset

- 2,499 Wall Street Journal (WSJ) stories
- Syntactic annotation
- Part-of-speech tagging

Model achieves F1 score of 92.22 on validation, 92.08 on test

- State-of-the-art performance (for a model that doesn't use external data or ensembling)

Experiments & Results

Output Correlations

Information about compatibility between outputs
in a structured prediction model

Parent Classification

Traditionally, parsers relied on correlations between predicted labels

Here, each label is scored independently, then dynamic programming chooses optimal tree

Hypothesis: LSTM handles reconciliation between labels that was previously handled by inference procedure

Parent Classification

Experiment: see if model can predict *parent* labels of spans

- Freeze input and LSTM parameters
- Train a new scoring network to predict the label of a span's parent instead of the label of the span

Result: able to achieve 92.3% accuracy, implying that information about parent-child relationships provided explicitly by a grammar is not lost

Independent Span Decisions

No explicit grammar, but inference stage does still enforce tree constraints

Experiment: remove these tree constraints and make decisions independently

Result: F1 score of 92.20, 94.5% of predictions form valid trees, implying that model learns tree constraints on its own

Lexical Representation

Importance of word and character representations

Alternate Word Representations

Experiment: ablation study

- Word embeddings alone are not sufficient
- Character representations and part-of-speech tags provide much of the same information
- Larger character LSTM can make up for loss of word embeddings

Word and Character LSTM	92.22
Word Only	91.44
Word and Tag	92.09
Word, Tag, and Character LSTM	92.24
Character Only	92.24

Table 1: Development F1 scores on section 22 of the Penn Treebank for different lexical representations.

Predicting Word Features

Past work shows that word shapes, suffixes, and other attributes are important, especially for rare or <UNK> tokens

Hypothesis: character LSTM learns similar information

Experiment:

- Treat output of character LSTM as fixed word encoding
- Train small feedforward network to predict binary word features

Result: 99.7+% accuracy in all cases, implying that character LSTM learns word roots, suffixes, etc.

Context in the Sentence LSTM

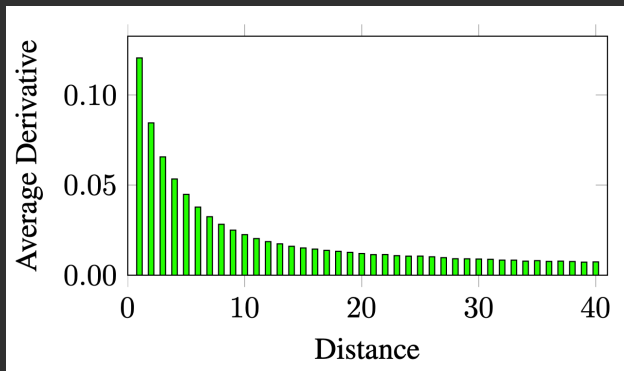
Importance of context

Derivative Analysis

Do LSTMs capture distant information?

Experiment: take gradient of each component of LSTM output vector w.r.t. each LSTM input vector

Result: even words 40 words away affect gradient



Truncation Analysis

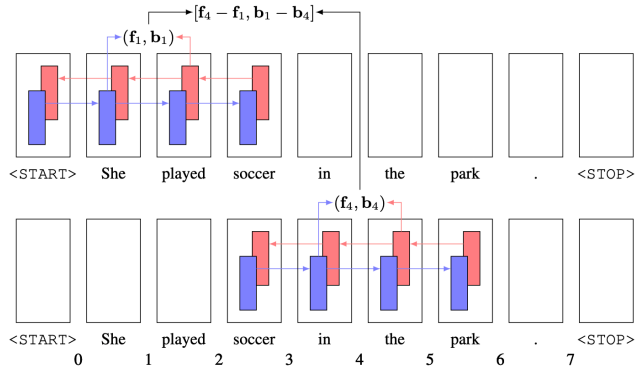


Figure 3: An example of creating a truncated span representation for the span "played soccer in" with context size $k = 2$. This representation is used to investigate the importance of information far away from the fenceposts of a span.

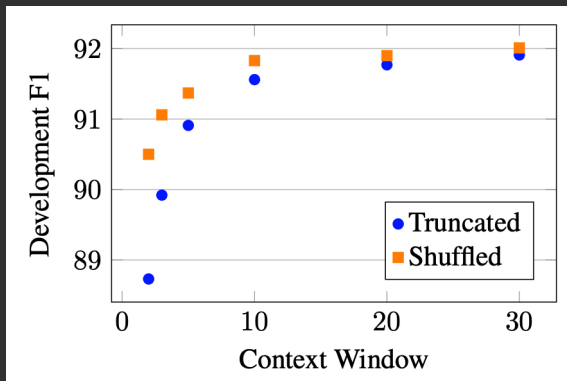
Truncation Analysis

How important is distant information?

Experiment:

- 1 Remove distant context info from span encoding
- 2 Add position-dependent cell state initialization

Result: distant context improves accuracy



Word Order

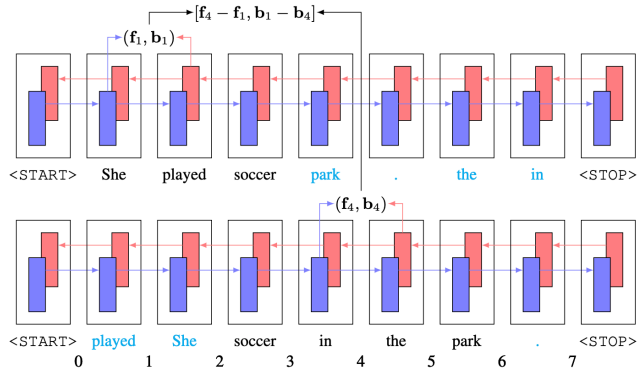


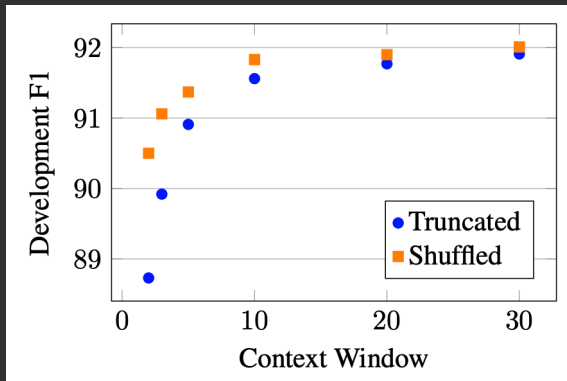
Figure 5: An example of creating a shuffled span representation for the span "played soccer in" with context size $k = 2$. The light blue words are outside the context window and are shuffled randomly. Shuffled representations are used to explore whether the order of far-away words is important.

Word Order

How important is distant word order?

Experiment: shuffle distant words to remove order info

Result: word order important, some context is better than no context



LSTMs vs. Feedforward



LSTMs vs. Feedforward

LSTMs can handle variable length sequences, but is that their only advantage?

Experiment:

- 1 Rerun truncation analysis experiment with feedforward network
- 2 Focus on context window of size $k = 3$
- 3 Concatenate learned position embedding to input

Feedforward network	83.39 F1
LSTM	89.92 F1

Discussion

Contributions

This paper provides important evidence for many different hypotheses about neural constituency parsing (and NLP in general)

- LSTMs are capable of learning parent-child relationships and tree constraints
- Character representations learn much of the same information that part-of-speech tags once provided
- Distant context, and even word order is important for LSTM performance
- LSTMs are superior to MLPs for NLP

Critique

Still a lot we don't know about how LSTMs learn and how knowledge is represented within the model

- In “predicting word features” experiment, don't compare against performance of word embeddings
- In “truncation analysis” experiment, don't use size-dependent cell state, and don't retune hyperparameters
- Only tested on English

Questions?

Appendix

Component	Dimensions	Layers
Word Embeddings	100	
Character Embeddings	50	
Character LSTM	100	1
Sentence LSTM	250	2
Label Feedforward Network	250	1




Table 2: The sizes of the components used in our model.

Appendix

Binary Feature	Majority Class	Char-LSTM Classifier	Binary Feature	Majority Class	Char-LSTM Classifier
all-letters	77.22%	99.77%	suffix = “s”	82.65%	99.99%
has-letter	89.18%	99.97%	suffix = “ed”	92.52%	99.98%
all-lowercase	56.95%	99.95%	suffix = “ing”	93.26%	99.95%
has-lowercase	85.85%	99.90%	suffix = “ion”	97.75%	99.93%
all-uppercase	96.68%	99.90%	suffix = “er”	96.42%	99.97%
has-uppercase	67.77%	99.97%	suffix = “est”	99.63%	99.98%
all-digits	98.38%	99.99%	suffix = “ly”	97.56%	99.99%
has-digit	87.90%	99.91%	suffix = “ity”	99.30%	99.94%
all-punctuation	99.93%	99.98%	suffix = “y”	92.97%	99.93%
has-punctuation	79.04%	99.75%	suffix = “al”	98.48%	99.92%
has-dash	88.89%	99.95%	suffix = “ble”	99.30%	99.90%
has-period	92.55%	99.95%	suffix = “e”	89.57%	99.99%
has-comma	98.02%	99.97%			

Table 3: Classification accuracy for various binary word features using the character LSTM representations for words induced by a pre-trained parser. Performance substantially exceeds that of a majority class classifier in all cases, reaching 99.7% or higher for all features. The majority class is `True` for the first four features in the left column and `False` for the rest.

References

-  Amidi, A. & Amidi, S. *Recurrent neural networks cheatsheet*. <https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks>. 2018.
-  Gaddy, D., Stern, M. & Klein, D. What's going on in neural constituency parsers? an analysis. *Proceedings of NAACL-HLT 2018*, 999–1010 (2018).
-  Jurafsky, D. & Martin, J. H. *Speech and language processing*. Third edition draft (2019).

What Do Recurrent Neural Network Grammars Learn About Syntax?

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong,
Chris Dyer, Graham Neubig, Noah A. Smith

Outline

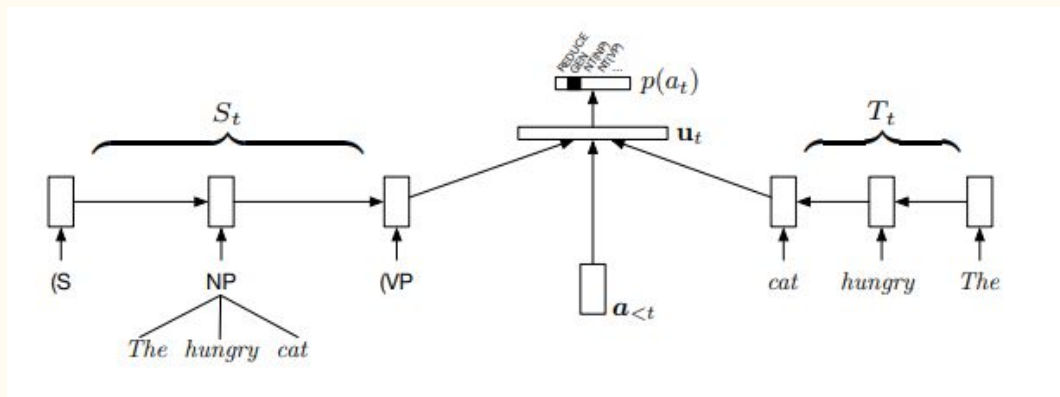
- RNNG
- Composition is Key
- Gated Attention RNNG
- Headedness in Phrases
- The Role of Nonterminal Labels
- Related Work

Recurrent Neural Network Grammars

- RNNG defines a joint probability distribution over string terminals and phrase-structure nonterminals.
- $\langle N, \Sigma, \Theta \rangle$
 - N : the set of nonterminal symbols (NP, VP, etc.)
 - Σ : the set of all terminal symbols
 - Θ : the set of all model parameters

Recurrent Neural Network Grammars

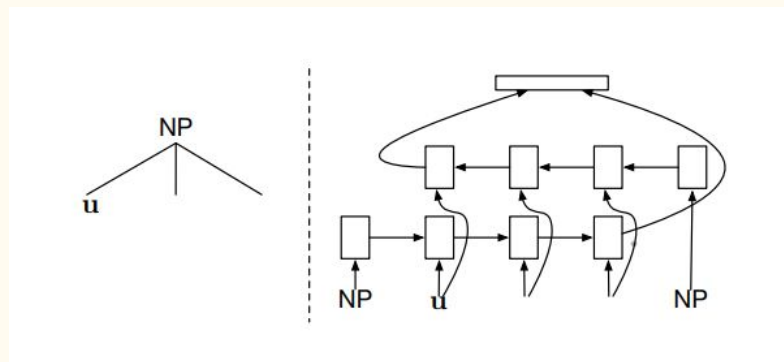
- Algorithmic state
 - Stack: partially completed constituents
 - Buffer: already-generated terminal symbols
 - List of past actions
- Phrase-structure tree y , sentence x
 - Top-down
 - Oracle, $a = \langle a_1, \dots, a_n \rangle$



This figure is due to Dyer et al. (2016)

Recurrent Neural Network Grammars

- Actions
 - NT(X): introduces an open nonterminal symbol onto the stack
 - GEN(x): generates a terminal symbol and places it on the stack and buffer
 - REDUCE: indicates a constituent is now complete. (popped→composition function→pushed)



This figure is due to Dyer et al. (2016)

Recurrent Neural Network Grammars

- Composition function
 - Computes a vector representation
 - LSTM
- Generative model

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{a}) = \prod_{t=1}^n p(a_t \mid a_1, \dots, a_{t-1}).$$

Composition is Key

- Crucial role in the generalization success

Model	F_1
Vinyals et al. (2015) – PTB only	88.3
Discriminative RNNG	91.2
Choe and Charniak (2016) – PTB only	92.6
Generative RNNG	93.3

Table 1: Phrase-structure parsing performance on PTB §23. All results are reported using single-model performance and without any additional data.

Composition is Key

- Ablated RNNGs
 - Conjecture: Stack which makes use of the composition function is critical to the performance
 - The stack-only results are the best published PTB results

Model	F_1
Vinyals et al. (2015) [†]	92.1
Choe and Charniak (2016)	92.6
Choe and Charniak (2016) [†]	93.8
Baseline RNNG	93.3
Ablated RNNG (no history)	93.2
Ablated RNNG (no buffer)	93.3
Ablated RNNG (no stack)	92.5
Stack-only RNNG	93.6
GA-RNNG	93.5

Table 2: Phrase-structure parsing performance on PTB §23. [†] indicates systems that use additional unparsed data (semisupervised). The GA-RNNG results will be discussed in §4.

Model	UAS	LAS
Kiperwasser and Goldberg (2016)	93.9	91.9
Andor et al. (2016)	94.6	92.8
Dozat and Manning (2016)	95.4	93.8
Choe and Charniak (2016) [†]	95.9	94.1
Baseline RNNG	95.6	94.4
Ablated RNNG (no history)	95.4	94.2
Ablated RNNG (no buffer)	95.6	94.4
Ablated RNNG (no stack)	95.1	93.8
Stack-only RNNG	95.8	94.6
GA-RNNG	95.7	94.5

Table 3: Dependency parsing performance on PTB §23 with Stanford Dependencies (De Marneffe and Manning, 2008). [†] indicates systems that use additional unparsed data (semisupervised).

Model	Test ppl. (PTB)
IKN 5-gram	169.3
LSTM LM	113.4
RNNG	105.2
Ablated RNNG (no history)	105.7
Ablated RNNG (no buffer)	106.1
Ablated RNNG (no stack)	113.1
Stack-only RNNG	101.2
GA-RNNG	100.9

Table 4: Language modeling: perplexity. IKN refers to Kneser-Ney 5-gram LM.

Gated Attention RNNG

- Linguistic Hypotheses
 - Individual lexical head or multiple heads?
- Gated Attention Composition
 - GA-RNNG: explicit attention mechanism and a sigmoid gate with multiplicative interactions
 - “Attention weight”, weighted sum
 - $\mathbf{a} = \text{softmax} \left([\mathbf{c}_1 \ \mathbf{c}_2 \ \cdots]^\top \mathbf{V} [\mathbf{u}; \mathbf{o}_{nt}] \right)$
 - $\mathbf{g} = \sigma (\mathbf{W}_1 \mathbf{t}_{nt} + \mathbf{W}_2 \mathbf{m} + \mathbf{b})$
 - $\mathbf{c} = \mathbf{g} \odot \mathbf{t}_{nt} + (1 - \mathbf{g}) \odot \mathbf{m}.$
- Experimental results:
 - Outperforms the baseline RNNG with all three structures present
 - Achieves competitive performance with the strongest, stack-only, RNNG variant

Headedness in Phrases

- The Heads that GA-RNNG Learns
 - Average perplexity of the attention vectors
 - Resemble the vector of one salient constituent, but not exclusively
 - How attention is distributed for the major nonterminal categories
 - NPs, VPs and PPs

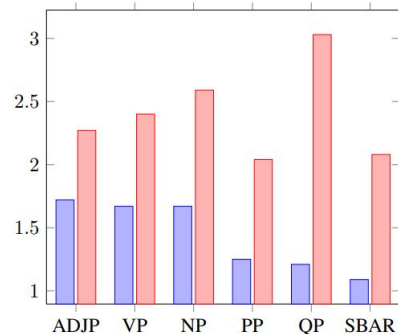



Figure 3: Average perplexity of the learned attention vectors on the test set (blue), as opposed to the average perplexity of the uniform distribution (red), computed for each major phrase type.

Noun phrases		Verb phrases	Prepositional phrases
1 Canadian (0.09) Auto (0.31) Workers (0.2) union (0.22) president (0.18)		buying (0.31) and (0.25) selling (0.21) NP (0.23)	ADVP (0.14) on (0.72) NP (0.14)
2 no (0.29) major (0.05) Eurobond (0.32) or (0.01) foreign (0.01) bond (0.1) offerings (0.22)		ADVP (0.27) show (0.29) PRT (0.23) PP (0.21)	ADVP (0.05) for (0.54) NP (0.40)
3 Saatchi (0.12) client (0.14) Philips (0.21) Lighting (0.24) Co. (0.29)		pleaded (0.48) ADJP (0.23) PP (0.15) PP (0.08) PP (0.06)	ADVP (0.02) because (0.73) of (0.18) NP (0.07)
4 nonperforming (0.18) commercial (0.23) real (0.25) estate (0.1) assets (0.25)		received (0.33) PP (0.18) NP (0.32) PP (0.17)	such (0.31) as (0.65) NP (0.04)
5 the (0.1) Jamaica (0.1) Tourist (0.03) Board (0.17) ad (0.20) account (0.40)		cut (0.27) NP (0.37) PP (0.22) PP (0.14)	from (0.39) NP (0.49) PP (0.12)
6 the (0.0) final (0.18) hour (0.81)		to (0.99) VP (0.01)	of (0.97) NP (0.03)
7 their (0.0) first (0.23) test (0.77)		were (0.77) n't (0.22) VP (0.01)	in (0.93) NP (0.07)
8 Apple (0.62) , (0.02) Compaq (0.1) and (0.01) IBM (0.25)		did (0.39) n't (0.60) VP (0.01)	by (0.96) S (0.04)
9 both (0.02) stocks (0.03) and (0.06) futures (0.88)		handle (0.09) NP (0.91)	at (0.99) NP (0.01)
10 NP (0.01) , (0.0) and (0.98) NP (0.01)		VP (0.15) and (0.83) VP (0.02)	NP (0.1) after (0.83) NP (0.06)

- Comparison to Existing Head Rules
 - Higher overlap with the conversion using Collins head rules rather than the Stanford head rules
 - GA-RNNG has the power to infer head rules

The Role of Nonterminal Labels

- Whether heads are sufficient to create representations of phrases
 - Unlabeled F_1 parsing accuracy: U-GA-RNNG 93.5%, GA-RNNG 94.2%
 - Visualization
 - Analysis of PP and SBAR
 - SBARs (start with “prepositional” words) are similar to PPs
 - The model learns to disregard the word *that*
 - Certain categories of PPs and SBARs form their own separate clusters
- 

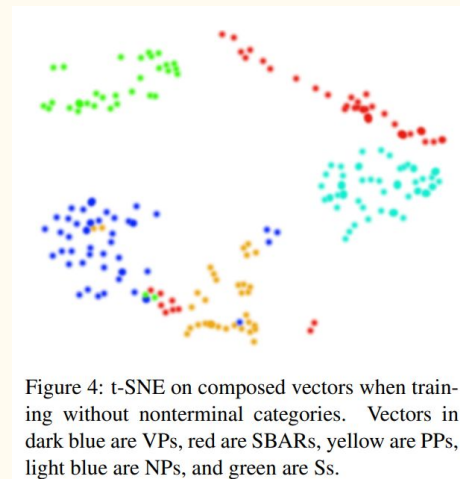
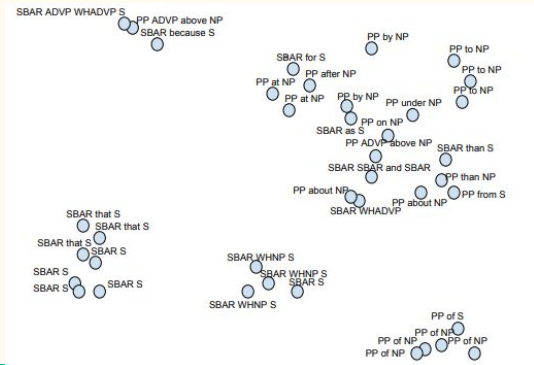


Figure 4: t-SNE on composed vectors when training without nonterminal categories. Vectors in dark blue are VPs, red are SBARs, yellow are PPs, light blue are NPs, and green are Ss.

Related Work

- Sequential RNNs (Karpathy et al., 2015; Li et al., 2016).
- Sequence-to sequence neural translation models capture a certain degree of syntactic knowledge of the source language as a by-product of the translation objective (Shi et al., 2016)
- Competitive parsing accuracy without explicit composition (Vinyals et al. ,2015; Wiseman and Rush, 2016)
- The importance of recursive tree structures in four different tasks (Li et al., 2015)
- The probabilistic context-free grammar formalism, with lexicalized (Collins, 1997) and nonterminal (Johnson, 1998; Klein and Manning, 2003) augmentations.
- Fine-grained nonterminal rules and labels can be discovered given weaker bracketing structures (Chiang and Bikel, 2002; Klein and Manning, 2002; Petrov et al., 2006)
- Entropy minimization and greedy familiarity maximization techniques to obtain lexical heads from labeled phrase-structure trees in an unsupervised manner (Sangati and Zuidema, 2009)

A teal square background with a white circle in the center. Inside the circle, the letters "NLP" are written in white, bold, sans-serif font.

NLP


Deep Semantic Role Labeling: What Works and What's Next

Luheng He, Kenton Lee, Mike Lewis, Luke Zettlemoyer

Presented by Huining Liang



Outline

- Model
 - Experiments
 - Analysis
 - Related Work
 - Conclusion and Future Work
- 

1

Model

- Predicting an SRL structure under our model involves finding the highest-scoring tag sequence over the space of all possibilities \mathcal{Y} :

$$\hat{\mathbf{y}} = \operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}} f(\mathbf{w}, \mathbf{y})$$

- Use a deep bidirectional LSTM (BiLSTM) to learn a locally decomposed scoring function conditioned on the input: $\sum_{t=1}^n \log p(y_t \mid \mathbf{w})$
- To incorporate additional information (e.g., structural consistency, syntactic input), we augment the scoring function with penalization terms:

$$f(\mathbf{w}, \mathbf{y}) = \sum_{t=1}^n \log p(y_t \mid \mathbf{w}) - \sum_{c \in \mathcal{C}} c(\mathbf{w}, y_{1:t})$$

1

Model

- Our model computes the distribution over tags using stacked BiLSTMs, which we define as follows:

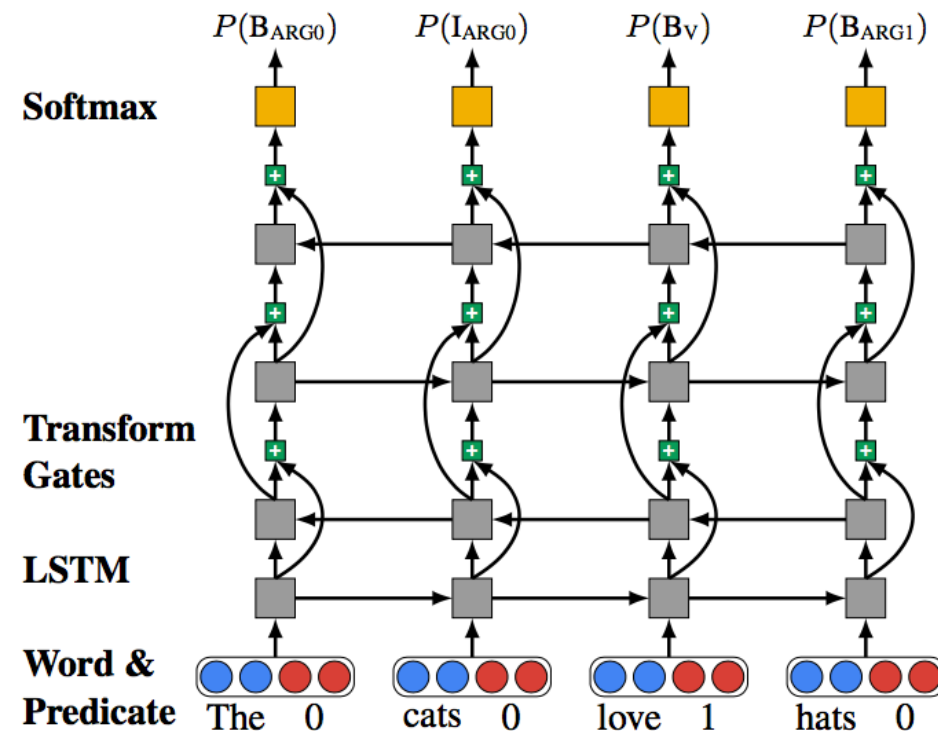
$$\begin{aligned} \mathbf{i}_{l,t} &= \sigma(\mathbf{W}_i^l[\mathbf{h}_{l,t+\delta_l}, \mathbf{x}_{l,t}] + \mathbf{b}_i^l) & \tilde{\mathbf{c}}_{l,t} &= \tanh(\mathbf{W}_c^l[\mathbf{h}_{l,t+\delta_l}, \mathbf{x}_{l,t}] + \mathbf{b}_c^l) \\ \mathbf{o}_{l,t} &= \sigma(\mathbf{W}_o^l[\mathbf{h}_{l,t+\delta_l}, \mathbf{x}_{l,t}] + \mathbf{b}_o^l) & \mathbf{c}_{l,t} &= \mathbf{i}_{l,t} \circ \tilde{\mathbf{c}}_{l,t} + \mathbf{f}_{l,t} \circ \mathbf{c}_{t+\delta_l} \\ \mathbf{f}_{l,t} &= \sigma(\mathbf{W}_f^l[\mathbf{h}_{l,t+\delta_l}, \mathbf{x}_{l,t}] + \mathbf{b}_f^l + 1) & \mathbf{h}_{l,t} &= \mathbf{o}_{l,t} \circ \tanh(\mathbf{c}_{l,t}) \end{aligned}$$

- To stack the LSTMs in an interleaving pattern, the layer-specific inputs $\mathbf{x}_{l,t}$ and directionality δ_l are arranged in the following manner:

$$\mathbf{x}_{l,t} = \begin{cases} [\mathbf{W}_{\text{emb}}(w_t), \mathbf{W}_{\text{mask}}(t = v)] & l = 1 \\ \mathbf{h}_{l-1,t} & l > 1 \end{cases} \quad \delta_l = \begin{cases} 1 & \text{if } l \text{ is even} \\ -1 & \text{otherwise} \end{cases}$$

- Finally, the locally normalized distribution over output tags is computed via a softmax layer:

$$p(y_t \mid \mathbf{x}) \propto \exp(\mathbf{W}_{\text{tag}}^y \mathbf{h}_{L,t} + \mathbf{b}_{\text{tag}})$$



Highway LSTM with four layers.

1

Model

· Highway Connections

$$\mathbf{r}_{l,t} = \sigma(\mathbf{W}_r^l[\mathbf{h}_{l,t-1}, \mathbf{x}_t] + \mathbf{b}_r^l)$$

$$\mathbf{h}'_{l,t} = \mathbf{o}_{l,t} \circ \tanh(\mathbf{c}_{l,t})$$

$$\mathbf{h}_{l,t} = \mathbf{r}_{l,t} \circ \mathbf{h}'_{l,t} + (1 - \mathbf{r}_{l,t}) \circ \mathbf{W}_h^l \mathbf{x}_{l,t}$$

· Recurrent Dropout

$$\tilde{\mathbf{h}}_{l,t} = \mathbf{r}_{l,t} \circ \mathbf{h}'_{l,t} + (1 - \mathbf{r}_{l,t}) \circ \mathbf{W}_h^l \mathbf{x}_{l,t}$$

$$\mathbf{h}_{l,t} = \mathbf{z}_l \circ \tilde{\mathbf{h}}_{l,t}$$

· Constrained A* Decoding

$$f(\mathbf{w}, y_{1:t}) = \sum_{i=1}^t \log p(y_i | \mathbf{w}) - \sum_{c \in \mathcal{C}} c(\mathbf{w}, y_{1:i})$$

$$g(\mathbf{w}, y_{1:t}) = \sum_{i=t+1}^n \max_{y_i \in \mathcal{T}} \log p(y_i | \mathbf{w})$$

· BIO Constraints

These constraints reject any sequence that does not produce valid BIO transitions.

· SRL Constraints

Unique core roles (U)

Continuation roles (C)

Reference roles (R)

· Predicate Detection

We propose a simple model for end-to-end SRL, where the system first predicts a set of predicate words \mathbf{v} from the input sentence \mathbf{w} . Then each predicate in \mathbf{v} is used as an input to argument prediction.

2

Experiments

- Datasets

CoNLL-2005 & CoNLL-2012

Following the train-development-test split for both

Using the official evaluation script from the CoNLL 2005 shared task for evaluation on both datasets

- Model Setup

Our network consists of 8 BiLSTM layers (4 forward LSTMs and 4 reversed LSTMs) with 300-dimensional hidden units, and a softmax layer for predicting the output distribution.

Initialization - Training - Ensembling - Constrained Decoding

2

Experiments

Method	Development				WSJ Test				Brown Test				Combined
	P	R	F1	Comp.	P	R	F1	Comp.	P	R	F1	Comp.	F1
Ours (PoE)	83.1	82.4	82.7	64.1	85.0	84.3	84.6	66.5	74.9	72.4	73.6	46.5	83.2
Ours	81.6	81.6	81.6	62.3	83.1	83.0	83.1	64.3	72.9	71.4	72.1	44.8	81.6
Zhou	79.7	79.4	79.6	-	82.9	82.8	82.8	-	70.7	68.2	69.4	-	81.1
FitzGerald (Struct.,PoE)	81.2	76.7	78.9	55.1	82.5	78.2	80.3	57.3	74.5	70.0	72.2	41.3	-
Täckström (Struct.)	81.2	76.2	78.6	54.4	82.3	77.6	79.9	56.0	74.3	68.6	71.3	39.8	-
Toutanova (Ensemble)	-	-	78.6	58.7	81.9	78.8	80.3	60.1	-	-	68.8	40.8	-
Punyakanok (Ensemble)	80.1	74.8	77.4	50.7	82.3	76.8	79.4	53.8	73.4	62.9	67.8	32.3	77.9

Experimental results on CoNLL 2005, in terms of precision (P), recall (R), F1 and percentage of completely correct predicates (Comp.). We report results of our best single and ensemble (PoE) model.

Method	Development				Test			
	P	R	F1	Comp.	P	R	F1	Comp.
Ours (PoE)	83.5	83.2	83.4	67.5	83.5	83.3	83.4	68.5
Ours	81.8	81.4	81.5	64.6	81.7	81.6	81.7	66.0
Zhou	-	-	81.1	-	-	-	81.3	-
FitzGerald (Struct.,PoE)	81.0	78.5	79.7	60.9	81.2	79.0	80.1	62.6
Täckström (Struct.)	80.5	77.8	79.1	60.1	80.6	78.2	79.4	61.8
Pradhan (revised)	-	-	-	-	78.5	76.6	77.5	55.8

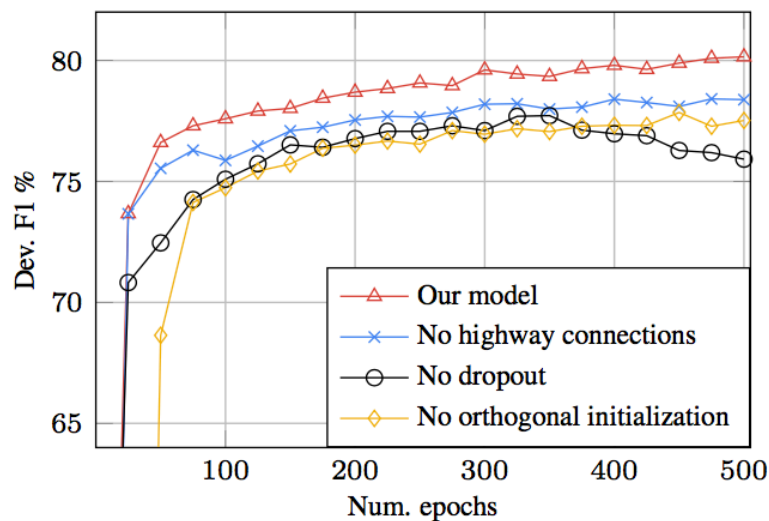
Experimental results on CoNLL 2012 in the same metrics as above. We compare our best single and ensemble (PoE) models against Zhou and Xu (2015), FitzGerald et al. (2015), Täckström et al. (2015) and Pradhan et al. (2013).

2

Experiments

Dataset	Predicate Detection			End-to-end SRL (Single)			End-to-end SRL (PoE)			
	P	R	F1	P	R	F1	P	R	F1	Δ F1
CoNLL 2005 Dev.	97.4	97.4	97.4	80.3	80.4	80.3	81.8	81.2	81.5	-1.2
WSJ Test	94.5	98.5	96.4	80.2	82.3	81.2	82.0	83.4	82.7	-1.9
Brown Test	89.3	95.7	92.4	67.6	69.6	68.5	69.7	70.5	70.1	-3.5
CoNLL 2012 Dev.	88.7	90.6	89.7	74.9	76.2	75.5	76.5	77.8	77.2	-6.2
CoNLL 2012 Test	93.7	87.9	90.7	78.6	75.1	76.8	80.2	76.6	78.4	-5.0

Predicate detection performance and end-to-end SRL results using predicted predicates. Δ F1 shows the absolute performance drop compared to our best ensemble model with gold predicates.



· Ablations

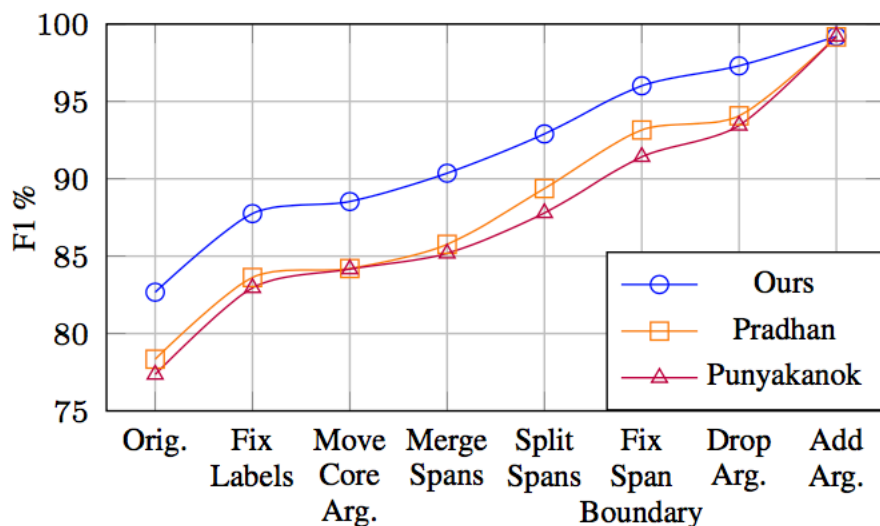
Smoothed learning curve of various ablations. The combination of highway layers, orthonormal parameter initialization and recurrent dropout is crucial to achieving strong performance. The numbers shown here are without constrained decoding.

3

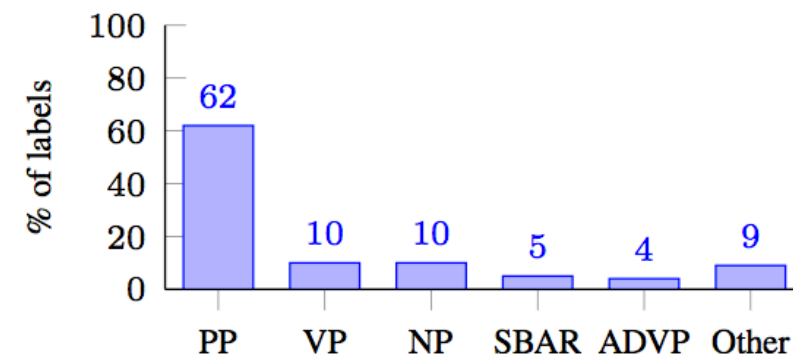
Analysis

· Error Types Breakdown

Label Confusion & Attachment Mistakes



Performance after doing each type of oracle transformation in sequence, compared to two strong non-neural baselines. The gap is closed after the Add Arg. transformation, showing how our approach is gaining from predicting more arguments than traditional systems.



For cases where our model either splits a gold span into two ($Z \rightarrow XY$) or merges two gold constituents ($XY \rightarrow Z$), we show the distribution of syntactic labels for the Y span. Results show the major cause of these errors is inaccurate prepositional phrase attachment.

3

Analysis

Operation	Description	%
Fix Labels	Correct the span label if its boundary matches gold.	29.3
Move Arg.	Move a unique core argument to its correct position.	4.5
Merge Spans	Combine two predicted spans into a gold span if they are separated by at most one word.	10.6
Split Spans	Split a predicted span into two gold spans that are separated by at most one word.	14.7
Fix Boundary	Correct the boundary of a span if its label matches an overlapping gold span.	18.0
Drop Arg.	Drop a predicted argument that does not overlap with any gold span.	7.4
Add Arg.	Add a gold argument that does not overlap with any predicted span.	11.0

Oracle transformations paired with the relative error reduction after each operation. All the operations are permitted only if they do not cause any overlapping arguments.

· Error Types Breakdown

Label Confusion & Attachment Mistakes

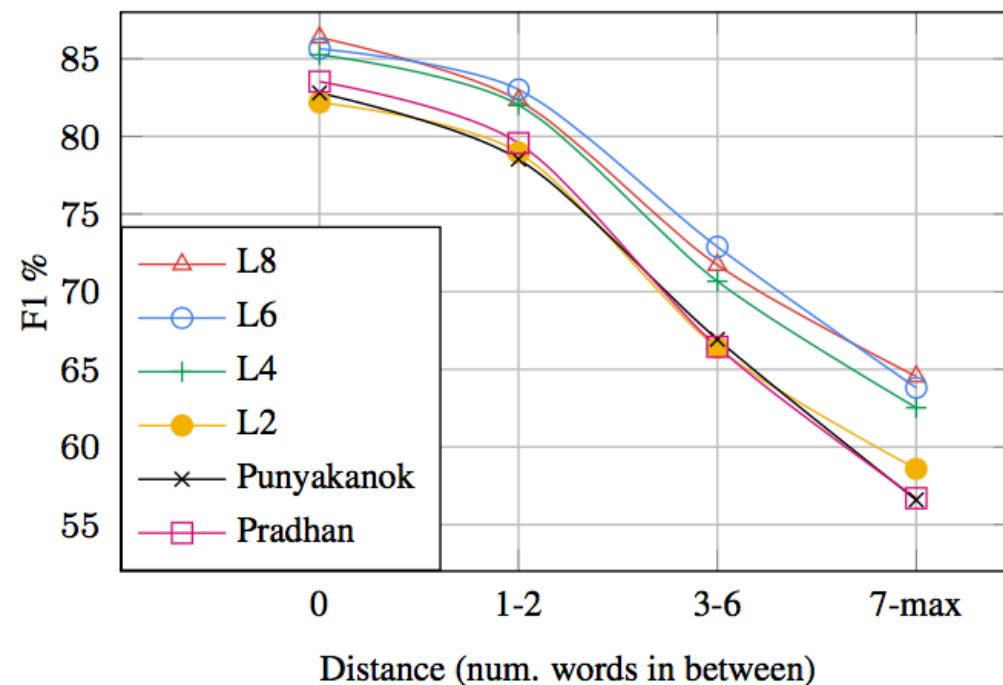
pred. \ gold	A0	A1	A2	A3	ADV	DIR	LOC	MNR	PNC	TMP
A0	-	55	11	13	4	0	0	0	0	0
A1	78	-	46	0	0	22	11	10	25	14
A2	11	23	-	48	15	56	33	41	25	0
A3	3	2	2	-	4	0	0	0	25	14
ADV	0	0	0	4	-	0	15	29	25	36
DIR	0	0	5	4	0	-	11	2	0	0
LOC	5	9	12	0	4	0	-	10	0	14
MNR	3	0	12	26	33	0	0	-	0	21
PNC	0	3	5	4	0	11	4	2	-	0
TMP	0	8	5	0	41	11	26	6	0	-

Confusion matrix for labeling errors, showing the percentage of predicted labels for each gold label. We only count predicted arguments that match gold span boundaries.

3

Analysis

- Long-range Dependencies



F1 by surface distance between predicates and arguments. Performance degrades least rapidly on long-range arguments for the deeper neural models.

3

Analysis

· Structural Consistency

BIO Violations & SRL Structure Violations

Model (no BIO)	Accuracy		Violations	Avg. Entropy	
	F1	Token	BIO	All	BIO
L8+PoE	81.5	91.5	0.07	0.02	0.72
L8	80.5	90.9	0.07	0.02	0.73
L6	80.1	90.3	0.06	0.02	0.72
L4	79.1	90.2	0.08	0.02	0.70
L2	74.6	88.4	0.18	0.03	0.66

Comparison of BiLSTM models without BIO decoding.

Gold	ARG1	V	ARG2	ARG3
	Housing starts	are expected to	quicken	a bit from August's pace
Pred.	ARG0	V	ARG2	ARG2
+SRL	ARG0	V	ARG1	ARG2

Example where performance is hurt by enforcing the constraint that core roles may only occur once (+SRL).

Model or Oracle	F1	Syn %	SRL-Violations		
			U	C	R
Gold	100.0	98.7	24	0	61
L8+PoE	82.7	94.3	37	3	68
L8	81.6	94.0	48	4	73
L6	81.4	93.7	39	3	85
L4	80.5	93.2	51	3	84
L2	77.2	91.3	96	5	72
L8+PoE+SRL	82.8	94.2	5	1	68
L8+PoE+AutoSyn	83.2	96.1	113	3	68
L8+PoE+GoldSyn	85.0	97.6	102	3	68
Punyakanok	77.4	95.3	0	0	0
Pradhan	78.3	93.0	84	3	58

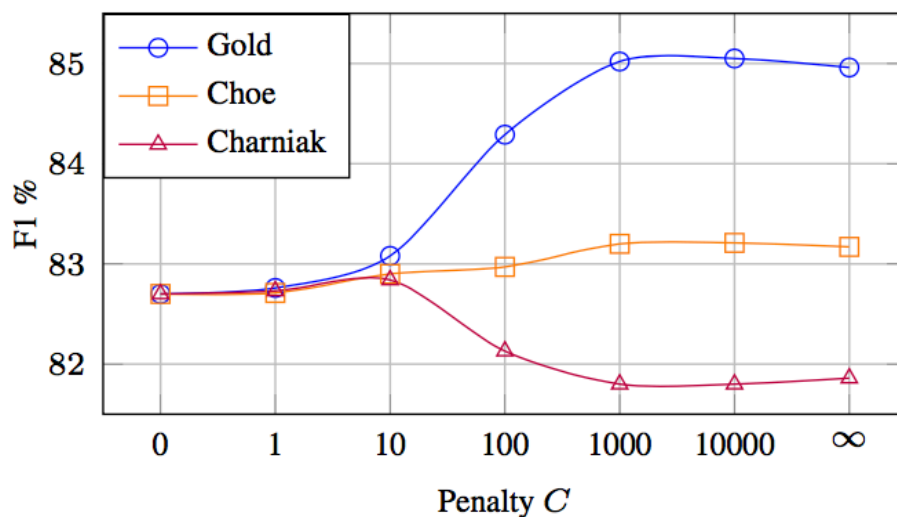
Comparison of models with different depths and decoding constraints (in addition to BIO) as well as two previous systems.

3

Analysis

· Can Syntax Still Help SRL?

Constrained Decoding with Syntax



Performance of syntax-constrained decoding as the non-constituent penalty increases for syntax from two parsers and gold syntax. The best existing parser gives a small improvement, but the improvement from gold syntax shows that there is still potential for syntax to help SRL.

	CoNLL-05		CoNLL-2012 Dev.						
	Dev.	Test	BC	BN	NW	MZ	PT	TC	WB
L8+PoE	82.7	84.6	81.4	82.8	82.8	80.4	93.6	84.8	81.0
+AutoSyn	83.2	84.8	81.5	82.8	83.2	80.6	93.7	84.9	81.1

F1 on CoNLL 2005, and the development set of CoNLL 2012, broken down by genres. Syntax-constrained decoding (+AutoSyn) shows bigger improvement on in-domain data (CoNLL 05 and CoNLL 2012 NW).

4

Related Work

- Traditional approaches to semantic role labeling have used syntactic parsers to identify constituents and model long-range dependencies, and enforced global consistency using integer linear programming (Punyakanok et al., 2008) or dynamic programs (Tačkstrõmetal.,2015).
- More recently, neural methods have been employed on top of syntactic features (FitzGerald et al., 2015; Roth and Lapata, 2016) .
- Our experiments show that off-the-shelf neural methods have a remarkable ability to learn long-range dependencies, syntactic constituency structure, and global constraints without coding task-specific mechanisms for doing so.

Conclusion and Future Work

- **A new deep learning model for span-based semantic role labeling**

10% relative error reduction over the previous state of the art

Ensemble of 8 layer BiLSTMs incorporated some of the recent best practices(orthonormal initialization, RNN-dropout, and highway connections, which are crucial for getting good results with deep models)

- **Extensive error analysis shows the strengths and limitations of our deep SRL model compared with shallower models and two strong non-neural systems.**

Our deep model is better at recovering long-distance predicate-argument relations

Structural inconsistencies(which can be alleviated by constrained decoding)

- **The question of whether deep SRL still needs syntactic supervision**

Despite recent success without syntactic input, there is still potential for high quality parsers to further improve deep SRL models.

References

- Claire Bonial, Olga Babko-Malaya, Jinho D Choi, Jena Hwang, and Martha Palmer. 2010. Propbank annotation guidelines. Center for Computational Language and Education Research Institute of Cognitive Science University of Colorado at Boulder .
- Xavier Carreras and Lluís Ma`rquez.2005. Introduction to the conll-2005 shared task: Semantic role labeling. In Proceedings of the Ninth Conference on Computational Natural Language Learning. Association for Computational Linguistics, pages 152–164.
- Eugene Charniak. 2000. A maximum-entropy-inspired parser. In Proc. of the First North American chapter of the Association for Computational Linguistics conference (NAACL). Association for Computational Linguistics, pages 132–139.
- Do Kook Choe and Eugene Charniak. 2016. Parsing as language modeling. In Proc. of the 2016 Conference of Empirical Methods in Natural Language Processing (EMNLP).