Deep Biaffine Attention Dependency Parsing

黃静娴

[Dozat and Manning 2017&2018]

Contents

Graph-based Dependency Parsing

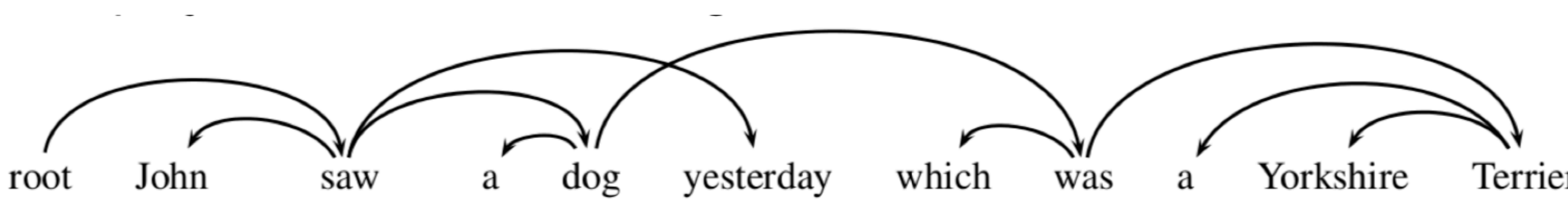
Graph-Based Dependency Parsing

$$G = < V, E >$$

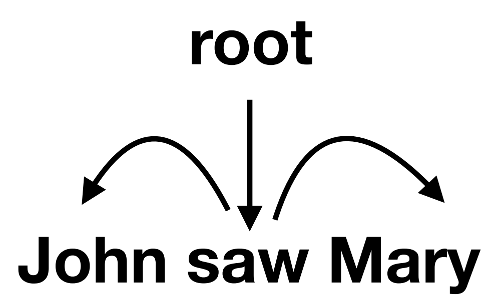
$$V = \{w_0 = root, w_1, \ldots, w_n\}$$

$$E = \{(w_r, r, w_j) : i \neq j, w_i \in V, w_j \in V - w_0, r \in R\}$$

$R =$ a set of all possible dependency relationship



Graph-Based Dependency Parsing

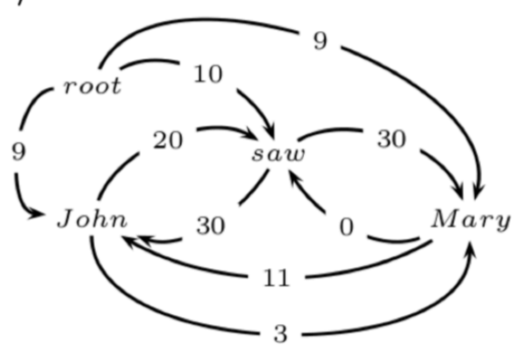


Q1. Between which of the two nodes do the edges exist?

Q2. What is the label of the edge?

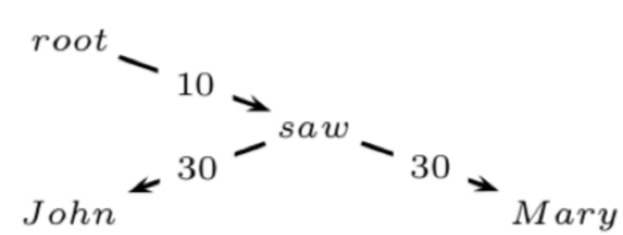
Graph-Based Dependency Parsing

$G = \langle V, E \rangle =$



($O(n^2)$ edges)

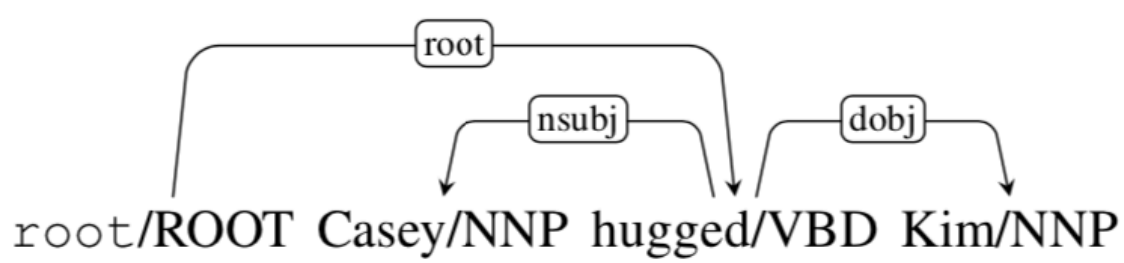
MST



Syntactic Parsing

Introduction

Syntax: provides rules to put together words to form components of sentence and to put together these components to form sentences.



Method

1. Arc score

Sentence 1: ***John saw Marry.***

For word w_i, the head word class include 4 words.

Sentence 2: ***Boeing is located in Seattle.***

For word w_i, the head word class include 6 words.

Traditional classifier: MLP

$$\mathbf{s}_i = W\mathbf{r}_i + \mathbf{b}$$

Fixed!

Methods

Traditional classifier: MLP

$$\mathbf{s}_i = W\mathbf{r}_i + \mathbf{b}$$

Fixed!

$\mathbf{s}_i$: the score for each word $\quad$ $\mathbf{r}_i$: the feature for each word

$$\mathbf{h}_i^{(arc-dep)} = MLP^{(arc-dep)}(\mathbf{r}_i)$$

$$\mathbf{h}_i^{(arc-head)} = MLP^{(arc-head)}(\mathbf{r}_i)$$

Methods

Traditional classifier: MLP

$$\mathbf{s}_i = W\mathbf{r}_i + \mathbf{b}$$

$$\mathbf{s}_i^{(arc)} = H^{(arc-head)}U^{(1)}\mathbf{h}_i^{(arc-dep)} + H^{(arc-head)}\mathbf{u}^{(2)}$$

Without stack $(d\times k)(k\times k)(k\times 1)+(d\times k)(k\times 1)=(d\times 1)$

With stack $(d\times k)(k\times k)(k\times d)+(d\times k)(k\times d)=(d\times d)$

Methods

2. Arc label

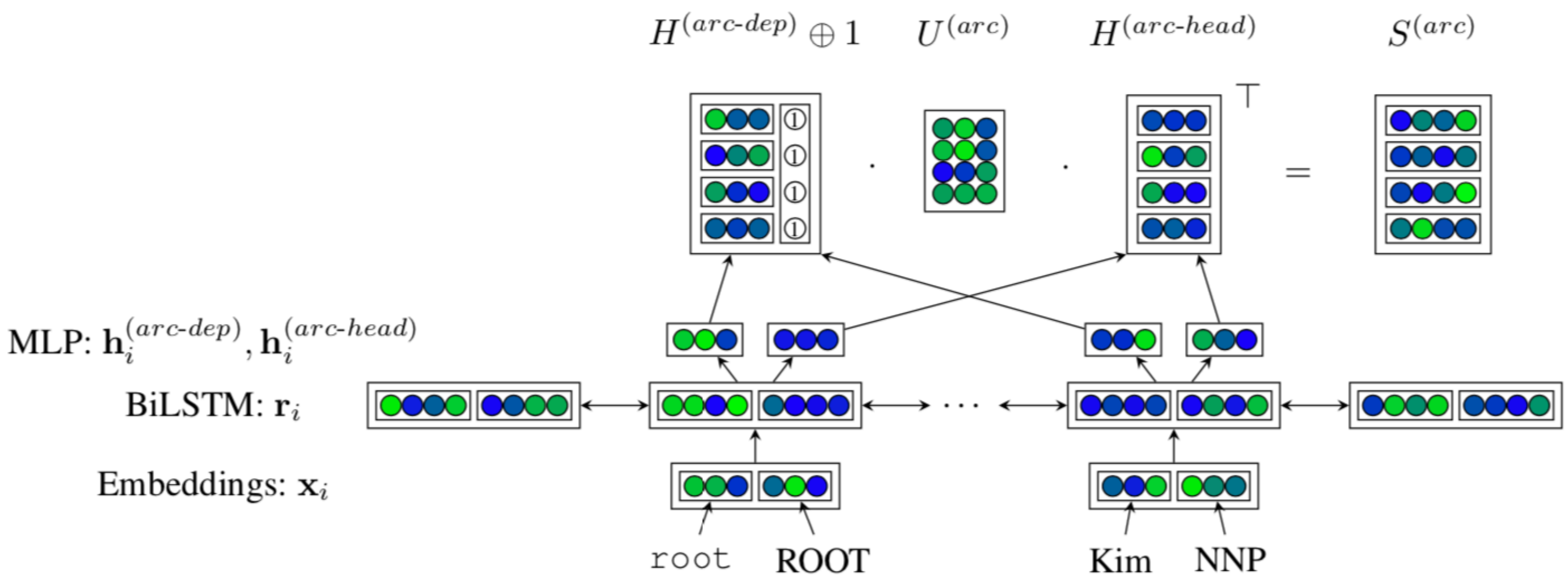
$$\mathbf{s}_i^{(label)} = \mathbf{r}_{y_i}^{\top} U^{(1)} \mathbf{r}_i + (\mathbf{r}_{y_i} \oplus \mathbf{r}_i)^{\top} U^{(2)} + \mathbf{b}$$

Fixed-class biaffine classifier

posterior probability with i being dep and y_i *being head*

posterior probability with i or y_i *being the end of an arc*

Methods



Model

Experiments

Hyperparameters

Param	Value	Param	Value
Embedding size	100	Embedding dropout	33%
LSTM size	400	LSTM dropout	33%
Arc MLP size	500	Arc MLP dropout	33%
Label MLP size	100	Label MLP dropout	33%
LSTM depth	3	MLP depth	1
α	$2e^{-3}$	β_1,β_2	.9
Annealing	$.75^{\frac{t}{5000}}$	t_{max}	50,000

Table 1: Model hyperparameters

Experiments

Dataset

Dataset	Tag
PTB-SD 3.3.0	Stanford POS Tag
PTB-SD 3.5.0	Stanford POS Tag
CTB	Gold Tag
CoNLL09	Provided predicted tag

Experiments

Result on PTB-SD 3.5.0

Classifier				Size			
Model	**UAS**	**LAS**	**Sents/sec**	**Model**	**UAS**	**LAS**	**Sents/sec**
Deep	**95.75**	**94.22**	**410.91**	3 layers, 400d	95.75	94.22	410.91
Shallow	95.74	94.00*	298.99	3 layers, 300d	95.82	**94.24**	460.01
Shallow, 50% drop	95.73	94.05*	300.04	3 layers, 200d	95.55*	93.89*	469.45
Shallow, 300d	95.63*	93.86*	373.24	2 layers, 400d	95.62*	93.98*	**497.99**
MLP	95.53*	93.91*	367.44	4 layers, 400d	**95.83**	94.22	362.09

Recurrent Cell			
Model	**UAS**	**LAS**	**Sents/sec**
LSTM	**95.75**	**94.22**	410.91
GRU	93.18*	91.08*	435.32
Cif-LSTM	95.67	94.06*	**463.25**

Experiments

Attention Mechanism

- deep bilinear model outperforms the others with respect to both speed and accuracy
- shallow bilinear arc and label classifiers
    - gets the same unlabeled performance as the deep model with the same settings
    - much slower and overfits
    - increasing the MLP dropout, doesn't change parsing speed
    - decrease the recurrent size to 300, hinders unlabeled accuracy without increasing parsing speed up to the same levels
- MLP of Kiperwasser & Goldberg(2016)
    - likewise be somewhat slower and significantly underperform the deep biaffine approach in both labeled and unlabeled accuracy.

Experiments

Recurrent cell

- GRU drastically underperformed LSTM
- Cif-LSTM(Coupled Input-forget Gate LSTM) slightly underperformed LSTM

Embedding dropout

-when one is dropped the other is scaled by a factor of two to compensate (overfitting)

-when both are dropped together, the model simply gets an input of zeros

- not using any tags at all actually results in better performance than using tags without dropout

Optimizer

Adam $\beta_2 = 0.9$

Experiments

Embedding dropout & Optimizer

Input Dropout		
Model	**UAS**	**LAS**
Default	**95.75**	**94.22**
No word dropout	95.74	94.08*
No tag dropout	95.28*	93.60*
No tags	95.77	93.91*

Adam		
Model	**UAS**	**LAS**
$\beta_2 = .9$	**95.75**	**94.22**
$\beta_2 = .999$	95.53*	93.91*

Experiments

Result

Type	Model	English PTB-SD 3.3.0 UAS	English PTB-SD 3.3.0 LAS	Chinese PTB 5.1 UAS	Chinese PTB 5.1 LAS
Transition	Ballesteros et al. (2016)	93.56	91.42	87.65	86.21
	Andor et al. (2016)	94.61	92.79	–	–
	Kuncoro et al. (2016)	**95.8**	**94.6**	–	–
Graph	Kiperwasser & Goldberg (2016)	93.9	91.9	87.6	86.1
	Cheng et al. (2016)	94.10	91.49	88.1	85.7
	Hashimoto et al. (2016)	94.67	92.90	–	–
	Deep Biaffine	95.74	94.08	**89.30**	**88.23**

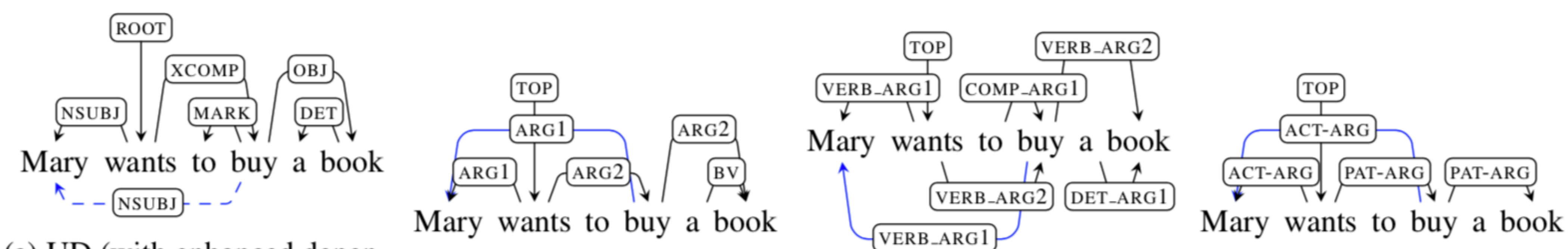
Table 4: Results on the English PTB and Chinese PTB parsing datasets

Semantic Parsing

Introduction

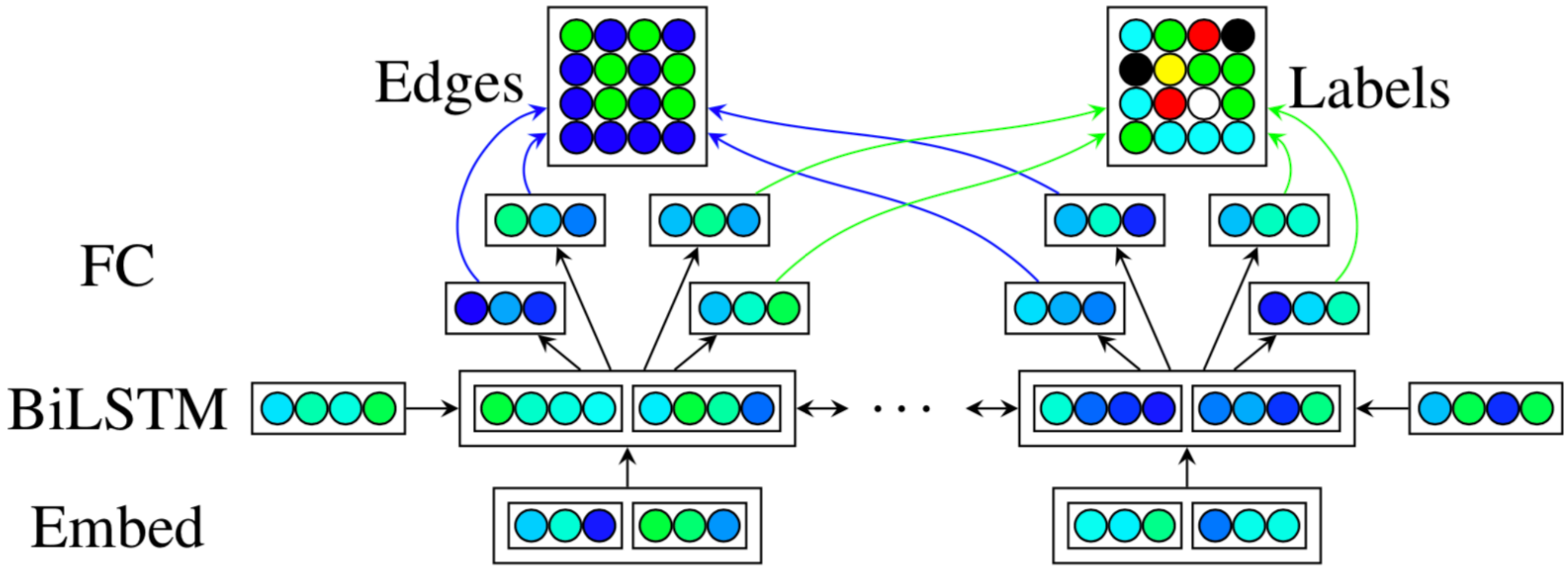
Syntactic parsing: tree structured

Semantic parsing: graph structured



(a) UD (with enhanced dependencies dashed) (b) DM (c) PAS (d) PSD

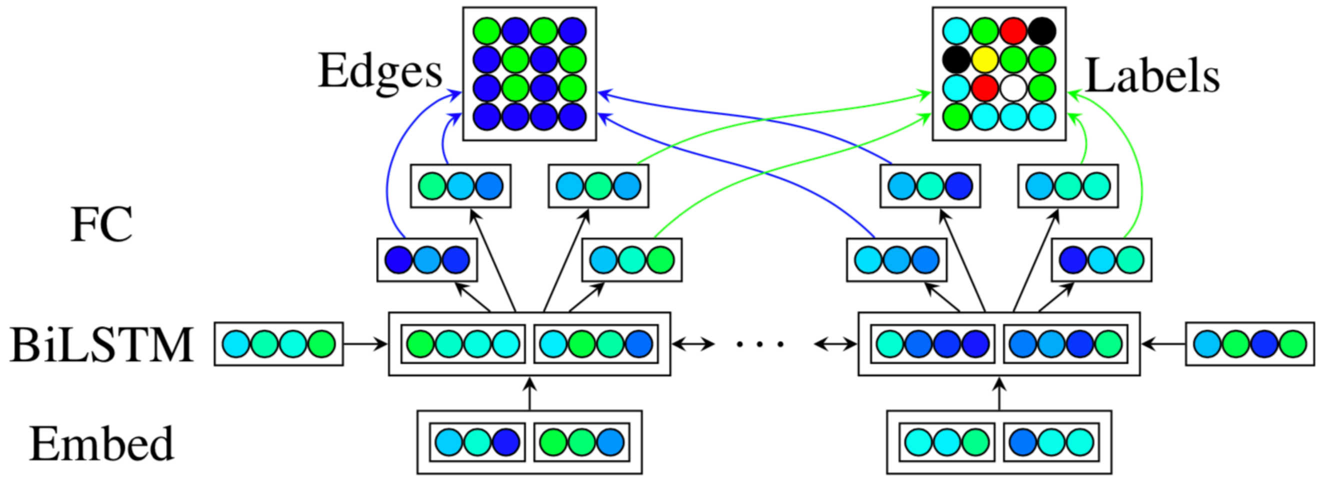
Methods



$$\mathbf{x}_i = \mathbf{e}_i^{(word)} \oplus \mathbf{e}_i^{(tag)}$$

$$R = BiLSTM(X)$$

Methods



$$\mathrm{Bilin}(\mathbf{v}_1, \mathbf{v}_2) = \mathbf{v}_1^T \mathbf{U} \mathbf{v}_2 + \mathbf{b}$$

$$\mathrm{Biaff}(\mathbf{v}_1, \mathbf{v}_2) = \mathbf{v}_1^T \mathbf{U} \mathbf{v}_2 + W(\mathbf{v}_1 \oplus \mathbf{v}_2) + \mathbf{b}$$

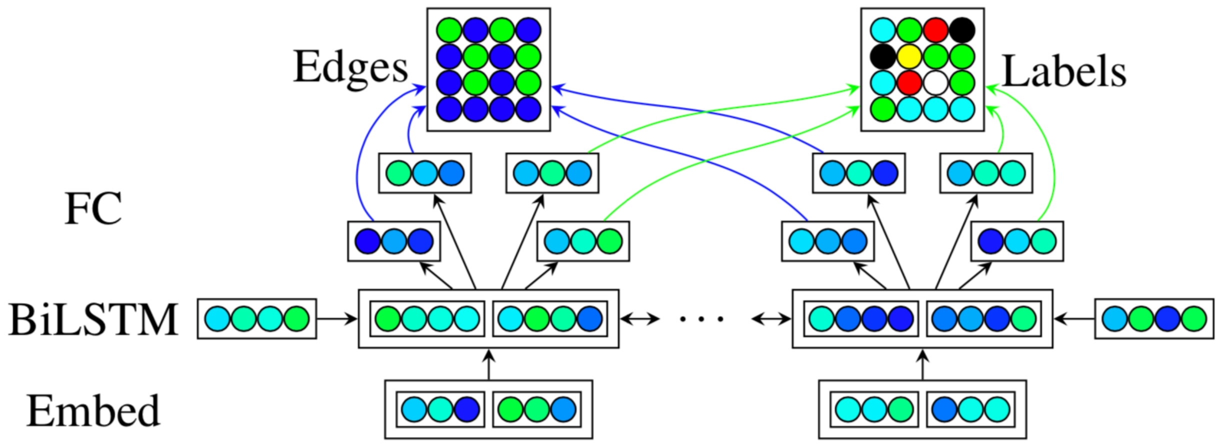
$$\mathbf{h}_i^{(edge-head)} = FNN^{(edge-head)}(\mathbf{r}_i)$$

$$\mathbf{h}_i^{(label-head)} = FNN^{(label-head)}(\mathbf{r}_i)$$

$$\mathbf{h}_i^{(edge-dep)} = FNN^{(edge-dep)}(\mathbf{r}_i)$$

$$\mathbf{h}_i^{(label-dep)} = FNN^{(label-dep)}(\mathbf{r}_i)$$

Methods



$$s_{ij}^{(edge)} = \text{Biaff}^{(edge)}(\mathbf{h}_i^{(edge-dep)}, \mathbf{h}_j^{(edge-head)})$$

$$s_{ij}^{(label)} = \text{Biaff}^{(label)}(\mathbf{h}_i^{(label-dep)}, \mathbf{h}_j^{(label-head)})$$

$$y_{i,j}^{'(edge)} = \{s_{i,j} \geq 0\}$$

$$y_{i,j}^{'(label)} = \text{argmax}\mathbf{s}_{i,j}$$

$$l = \lambda l^{(label)} + (1 - \lambda) l^{(edge)}$$

Experiments

Hyperparameter

Hidden Sizes	
Word/Glove/POS/ Lemma/Char	100
GloVe linear	125
Char LSTM	1 @ 400
Char linear	100
BiLSTM	3 @ 600
Arc/Label	600
Dropout Rates (drop prob)	
Word/GloVe/ POS/Lemma	20%
Char LSTM (FF/recur)	33%
Char linear	33%
BiLSTM (FF/recur)	45%/25%
Arc/Label	25%/33%
Loss & Optimizer	
Interpolation (λ)	.025
L_2 regularization	$3e^{-9}$
Learning rate	$1e^{-3}$
Adam β_1	0
Adam β_2	.95

Experiments

Dataset

- DM: The reduction of Minimal Recursion Semantics, available through the HPSG annotation of the WSJ text, into bi-lexical dependencies (Flickinger, et al., 2012).
- PAS: Predicate-Argument Structures extracted from another HPSG annotation of the PTB phrase structure trees (Miyao, et al., 2004).
- PSD: A reduction of the tectogrammatical analysis layer of the Prague Czech-English Dependency Treebank (Cinková, et al., 2009).

Experiments

Result

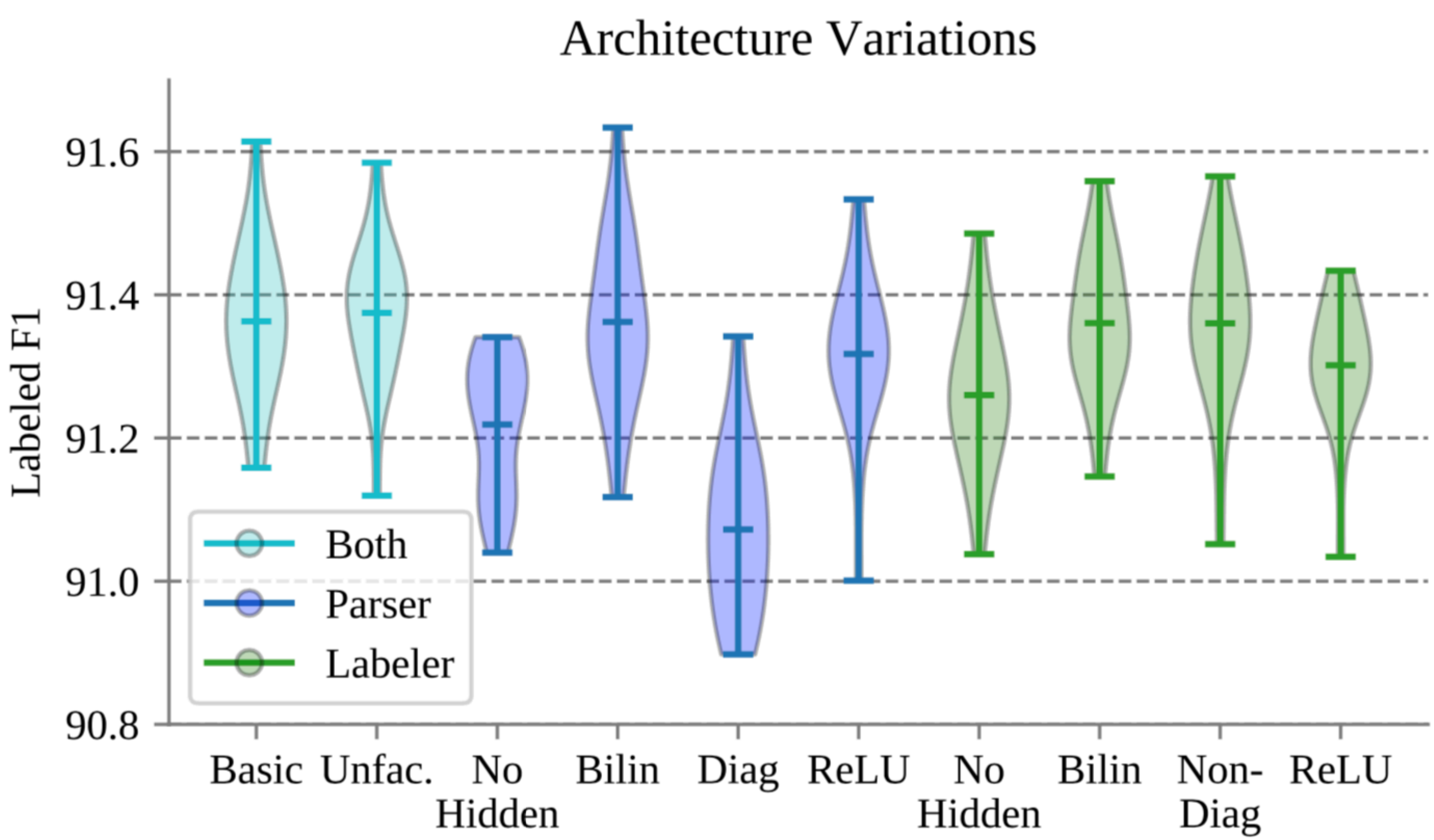
	DM		PAS		PSD		Avg	
	ID	OOD	ID	OOD	ID	OOD	ID	OOD
(Du et al., 2015)	89.1	81.8	91.3	87.2	75.7	73.3	85.3	80.8
(Almeida and Martins, 2015)	88.2	81.8	90.9	86.9	76.4	74.8	85.2	81.2
WCGL18	90.3	84.9	91.7	87.6	78.6	75.9	86.9	82.8
PTS17: Basic	89.4	84.5	92.2	88.3	77.6	75.3	87.4	83.6
PTS17: Freda3	90.4	85.3	92.7	89.0	78.5	76.4	88.0	84.4
Ours: Basic	91.4	86.9	93.9	**90.8**	79.1	77.5	88.1	85.0
Ours: +Char	92.7	87.8	**94.0**	90.6	80.5	78.6	89.1	85.7
Ours: +Lemma	93.3	88.8	93.9	90.5	80.3	78.7	89.1	86.0
Ours: +Char +Lemma	**93.7**	**88.9**	93.9	90.6	**81.0**	**79.4**	**89.5**	**86.3**

PTS17: Basic represents the single-task versions of Peng et al. (2017), which they make multitask across the three datasets in Freda3 by adding frustratingly easy domain adaptation and a third-order decoding mechanism.

WCGL18 is Wang et al.'s (2018) transition-based system.

Experiments

Architecture Variance



Summary

- Deep biaffine attention can be applied to both syntactic parsing and semantic parsing.

- Simplify the decoding structure without decrease the performance.