School of Information Science and Technology
ShanghaiTech University

Bidirectional Transition-Based Dependency Parsing

Yunzhe Yuan

Yunzhe Yuan, Yong Jiang, Kewei Tu
{yuanyzh,jiangyong,tukw}@shanghaitech.edu.cn

January 12, 2019

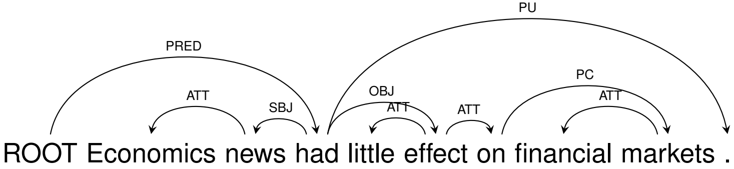
Overview

Background

Bidirectional Transition-Based Parsing
- Vanilla Joint Scoring
- Joint Decoding with Dual Decomposition
- Joint Decoding Guided by Dynamic Oracle

Experiments

Dependency Parsing



Transition-Based Dependency Parsing

Arc-hybrid System

- configuration $c = (\sigma, \beta, T)$
  - σ - stack
  - β - buffer
  - T - arc set
- action a

$$\mathrm{SHIFT}[(\sigma, b_0|\beta, T)] = (\sigma|b_0, \beta, T)$$
$$\mathrm{LEFT}_l[(\sigma|s_1|s_0, b_0|\beta, T)] = (\sigma|s_1, b_0|\beta, T \cup \{(b_0, s_0, l)\})$$
$$\mathrm{RIGHT}_l[(\sigma|s_1|s_0, \beta, T)] = (\sigma|s_1, \beta, T \cup \{(s_1, s_0, l)\})$$

ROOT Economics news had little effect on financial markets .

Action:
Stack:
Buffer: ROOT Economics news had little effect on financial markets .

ROOT Economics news had little effect on financial markets .

Action: SHIFT
Stack: ROOT
Buffer: Economics news had little effect on financial markets .

ROOT Economics news had little effect on financial markets .

Action: SHIFT
Stack: ROOT Economics
Buffer: news had little effect on financial markets .

Transition-Based Dependency Parsing
Example

ATT

ROOT Economics news had little effect on financial markets .

Action: LEFT_ATT
Stack: ROOT
Buffer: news had little effect on financial markets .

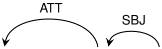
Transition-Based Dependency Parsing
Example

$$\text{ROOT Economics } \overset{\curvearrowleft \text{ ATT}}{\text{news}} \text{ had little effect on financial markets .}$$

Action: SHIFT
Stack: ROOT news
Buffer: had little effect on financial markets .

Transition-Based Dependency Parsing

Example



ROOT Economics news had little effect on financial markets .

Action: LEFT_SBJ
Stack: ROOT
Buffer: had little effect on financial markets .

Transition-Based Dependency Parsing
Example

ATT SBJ

ROOT Economics news had little effect on financial markets .

Action: SHIFT
Stack: ROOT had
Buffer: little effect on financial markets .

Transition-Based Dependency Parsing

Example

ATT SBJ

ROOT Economics news had little effect on financial markets .

Action: SHIFT

Stack: ROOT had little

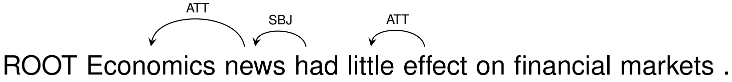
Buffer: effect on financial markets .

Transition-Based Dependency Parsing
Example

ATT SBJ ATT

ROOT Economics news had little effect on financial markets .

Action: LEFT_ATT
Stack: ROOT had
Buffer: effect on financial markets .



Action: SHIFT
Stack: ROOT had effect
Buffer: on financial markets .

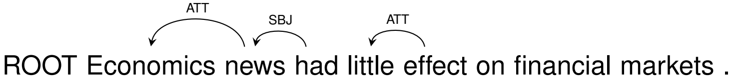
Transition-Based Dependency Parsing
Example

ATT SBJ ATT

ROOT Economics news had little effect on financial markets .

Action: SHIFT
Stack: ROOT had effect on
Buffer: financial markets .

Transition-Based Dependency Parsing
Example



Action: SHIFT
Stack: ROOT had effect on financial
Buffer: markets .

Transition-Based Dependency Parsing

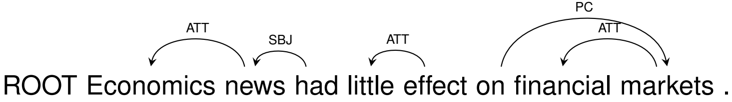
Example

ATT SBJ ATT ATT

ROOT Economics news had little effect on financial markets .

Action: LEFT_ATT
Stack: ROOT had effect on
Buffer: markets .

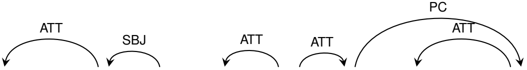
Transition-Based Dependency Parsing
Example



Action: RIGHT_PC
Stack: ROOT had effect
Buffer: on .

Transition-Based Dependency Parsing

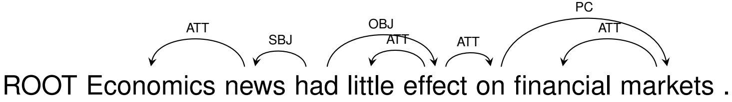
Example



ROOT Economics news had little effect on financial markets .

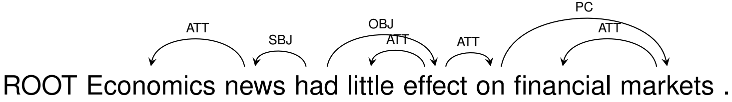
Action: RIGHT_ATT
Stack: ROOT had
Buffer: effect .

Transition-Based Dependency Parsing
Example



Action: RIGHT_OBJ
Stack: ROOT
Buffer: had .

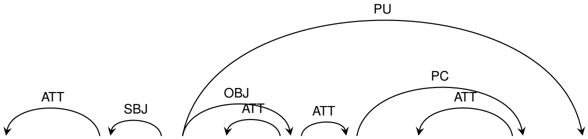
Transition-Based Dependency Parsing
Example



Action: SHIFT
Stack: ROOT had
Buffer: .

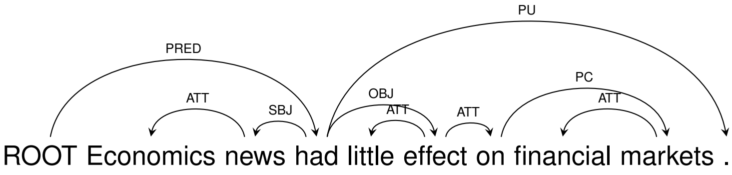
Transition-Based Dependency Parsing

Example

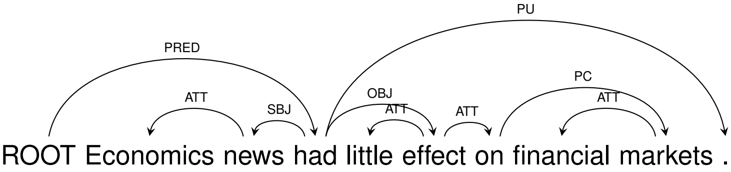


ROOT Economics news had little effect on financial markets .

Action: RIGHT_PU
Stack: ROOT
Buffer: had



Action: RIGHT_PRED
Stack:
Buffer: ROOT



Action: SHIFT
Stack: ROOT
Buffer:

Transition-Based Dependency Parsing

Algorithm

- Use the greedy algorithm
- Scoring function: MLP
- Input: Stack representation + Buffer representation
- Neural Net: BiLSTM

Input: a sentence $\mathbf{x}$

Output: a dependency parse tree $\mathbf{y}$

$c \leftarrow \mathrm{Initial}(\mathbf{x})$
while not Terminal(c) **do**
    $\hat{a} \leftarrow \arg\max_{a \in \mathrm{Legal}(c)} f(c, a)$
    $c \leftarrow \hat{a}(c)$
$\mathbf{y} \leftarrow c.T$
return $\mathbf{y}$

Motivation

- Traditionally the left-to-right manner is used
  $\sigma_0 = \varnothing, b_0 = \{\mathrm{ROOT}, \mathrm{Economics}, \mathrm{news}, \cdots, \mathrm{markets}, .\}$
- But the right-to-left manner also works
  $\sigma_0 = \varnothing, b_0 = \{\mathrm{ROOT}, ., \mathrm{markets}, \cdots, \mathrm{news}, \mathrm{Economics}\}$
- These two manners give different results

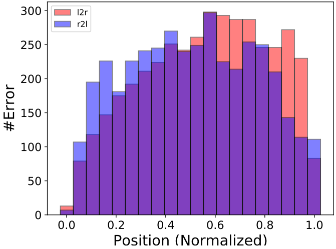
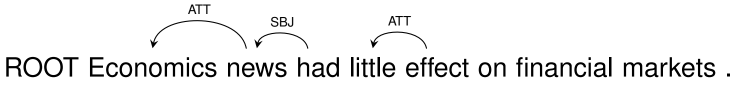


Figure: Comparison between results from the two bidirectional models

Motivation

- An early prediction mistake may negatively impact many future decision
  Correct:



  Wrong:

- All arcs after the third can be wrong
- Solution
  - Train parsers in both directions
  - Use the two parsers to do joint decoding

Bidirectional Parsing

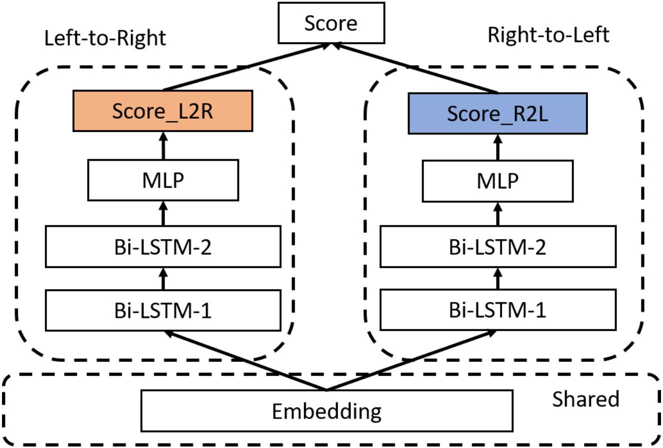


Figure: The structure of the bidirectional model

Vanilla Joint Scoring

- Simply add scores of two models together

$$s(\mathbf{t}) = F(\mathbf{t}) + G(\mathbf{t})$$

- Get two trees from the models respectively

$$\mathbf{y}, \mathbf{z}$$

- Select a better one according to the joint score

$$\arg\max_{\mathbf{t} \in \{\mathbf{y}, \mathbf{z}\}} s(\mathbf{t})$$

Joint Decoding with Dual Decomposition

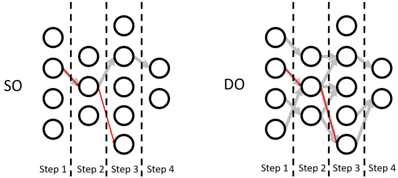
- The joint score can be written as

$$\arg\max_{\mathbf{y},\mathbf{z}} F(\mathbf{y}) + G(\mathbf{z})$$

  where $\mathbf{y} = \mathbf{z}$
- Use the iterative algorithm
  - Modify F and G to penalize dependencies of disagreement

Joint Decoding Guided by Dynamic Oracle

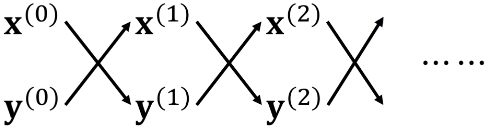
- Static oracle (SO) and Dynamic oracle (DO)



  - black circles: configurations
  - gray arrows: actions chosen when training
  - red arrows: actions chosen when decoding
- Benefit
  - Provide optimal actions for almost any configurations
  - Help the parser return to the gold parse in the fastest possible way

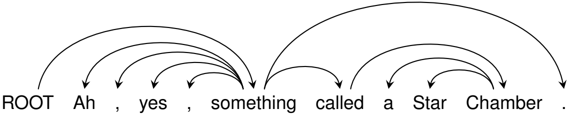
Joint Decoding Guided by Dynamic Oracle
Algorithm

- Joint decoding
  - Use the iterative algorithm
  - In each iteration, use the parse of one parser to construct a dynamic oracle that guides the other parser

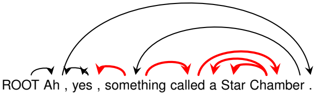


  - Modify the scoring function to encourage the parse tree to approach to its guide
- Comparison with dual decomposition
  - At least one action in each valid configuration would have its score modified
  - Dynamic oracles have more changes to the scoring function

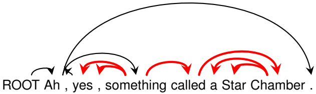
Iteration 0



Gold

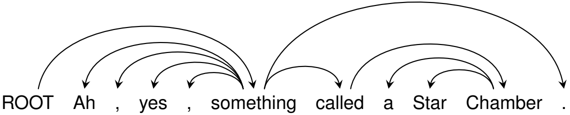


Left to right

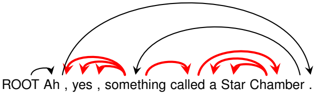


Right to left

Iteration 1



Gold

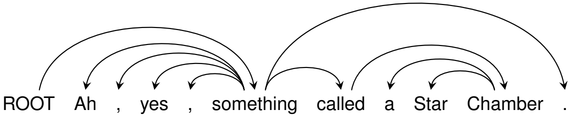


Left to right

Right to left

Joint Decoding Guided by Dynamic Oracle
Example

Iteration 2



Gold

Left to right

Right to left

Result

PTB and CTB

Method	PTB UAS	PTB LAS	CTB UAS	CTB LAS
L2R	93.54± 0.12	92.22± 0.17	86.21± 0.14	85.02± 0.13
R2L	93.56± 0.18	93.27± 0.25	86.44± 0.07	85.22± 0.07
Vanilla	94.35± 0.05	92.91± 0.11	87.36± 0.07	86.07± 0.06
DD	94.35± 0.05	93.01± 0.09	87.41± 0.09	86.18± 0.09
DO	**94.60± 0.04**	**94.02± 0.13**	88.07± 0.07	**87.54± 0.14**
DD + DO	**94.60± 0.04**	**94.02± 0.13**	88.09± 0.08	87.52± 0.10
C&M14	91.80	89.60	83.90	82.40
Dyer15	93.10	90.90	87.20	85.70
Weiss15	93.99	92.05	-	-
Andor16	**94.61**	92.79	-	-
Ballesteros16	93.56	91.42	87.65	86.21
K&G16	93.90	91.90	87.60	86.10
Zhang16	94.10	91.90	87.84	86.15
Shi17	94.53± 0.05	-	**88.62± 0.09**	-

Table: Results on PTB and CTB

Result

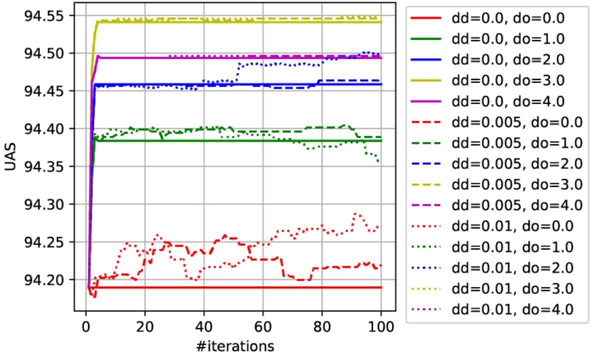
UD (without tuning)

Method	DE		EN		ES		FR	
	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
L2R	81.62	76.14	88.87	86.79	86.52	82.90	87.33	83.17
R2L	81.54	76.03	89.13	87.10	86.78	83.05	87.63	83.57
Vanilla	82.62	76.90	90.20	88.02	87.49	83.60	88.25	84.04
DD	82.64	77.12	90.23	88.24	87.52	83.78	88.30	84.77
DO	**83.02**	**79.58**	**90.56**	**89.48**	**87.83**	**85.69**	**88.81**	**87.82**

Method	IT		NL		PL		ZH	
	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
L2R	91.41	89.25	87.07	83.43	94.77	92.98	85.16	82.64
R2L	91.46	89.33	87.74	84.44	95.39	93.81	86.01	83.26
Vanilla	92.19	89.90	88.56	84.72	95.94	93.96	87.04	84.24
DD	92.22	90.61	88.58	85.04	95.96	94.47	87.06	84.38
DO	**92.31**	**91.58**	**89.41**	**87.41**	**96.10**	**94.62**	**87.75**	**86.46**

Table: Results on UD

Hyperparameter Tuning



Statistics

- Error distribution

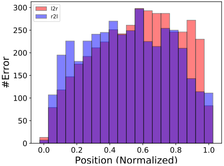


Figure: l2r and r2l

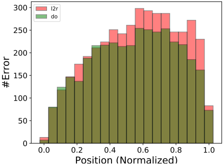


Figure: l2r and DO

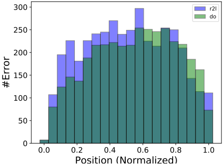


Figure: r2l and DO

Thank you!