Recurrent Neural Network Grammars

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, Noah A. Smith
Carnegie Mellon University, Pompeu Fabra University,
University of Washington

NAACL 2016

豊田工業大学 知能数理研究室 修士 1 年 外山洋太
第 8 回最先端 NLP 勉強会

導入

Recurrent Neural Network Grammars (RNNGs)

- Recurrent Neural Network (RNN) による文法のモデル
  - 単語や句の入れ子的・階層的構造を陽に表現
- 構文解析または文生成のアルゴリズムを用いて学習・推定
- タスク：構文解析，言語モデル
- 関連研究
  - Sequential な Recurrent Neural Networks (RNNs) は
    自然言語の潜在的な入れ子構造を考慮できていない
    → Recursive NN の手法を取り入れる
  - 既存の NN による手法はボトムアップ型（左隅）構文解析
    → 生成に適したトップダウン型のアルゴリズム

提案手法

RNNG の形式的な定義

$$RNNG := (N, \Sigma, \Theta)$$

$$\begin{cases} N : \text{非終端記号の有限集合} \\ \Sigma : \text{終端記号の有限集合} (N \cup \Sigma = \emptyset) \\ \Theta : \text{NN のパラメータ} \end{cases}$$

提案手法

構文解析のアルゴリズム

Stack$_t$	**Buffer**$_t$	**Open NTs**$_t$	**Action**	**Stack**$_{t+1}$	**Buffer**$_{t+1}$	**Open NTs**$_{t+1}$
S	B	n	NT(X)	$S \mid (\mathrm{X}$	B	$n+1$
S	$x \mid B$	n	SHIFT	$S \mid x$	B	n
$S \mid (\mathrm{X} \mid \tau_1 \mid \ldots \mid \tau_\ell$	B	n	REDUCE	$S \mid (\mathrm{X}\ \tau_1\ \ldots\ \tau_\ell)$	B	$n-1$

Input: *The hungry cat meows .*

	Stack	**Buffer**	**Action**
0		*The* | *hungry* | *cat* | *meows* | .	NT(S)
1	(S	*The* | *hungry* | *cat* | *meows* | .	NT(NP)
2	(S | (NP	*The* | *hungry* | *cat* | *meows* | .	SHIFT
3	(S | (NP | *The*	*hungry* | *cat* | *meows* | .	SHIFT
4	(S | (NP | *The* | *hungry*	*cat* | *meows* | .	SHIFT
5	(S | (NP | *The* | *hungry* | *cat*	*meows* | .	REDUCE
6	(S | (NP *The hungry cat*)	*meows* | .	NT(VP)
7	(S | (NP *The hungry cat*) | (VP	*meows* | .	SHIFT
8	(S | (NP *The hungry cat*) | (VP *meows*	.	REDUCE
9	(S | (NP *The hungry cat*) | (VP *meows*)	.	SHIFT
10	(S | (NP *The hungry cat*) | (VP *meows*) | .		REDUCE
11	(S (NP *The hungry cat*) (VP *meows*) .)		

提案手法

文生成のアルゴリズム

Stack$_t$	**Terms**$_t$	**Open NTs**$_t$	**Action**	**Stack**$_{t+1}$	**Terms**$_{t+1}$	**Open NTs**$_{t+1}$
S	T	n	NT(X)	$S \mid (\mathrm{X}$	T	$n+1$
S	T	n	GEN(x)	$S \mid x$	$T \mid x$	n
$S \mid (\mathrm{X} \mid \tau_1 \mid \ldots \mid \tau_\ell$	T	n	REDUCE	$S \mid (\mathrm{X}\ \tau_1\ \ldots\ \tau_\ell)$	T	$n-1$

	Stack	**Terminals**	**Action**
0			NT(S)
1	(S		NT(NP)
2	(S | (NP		GEN(*The*)
3	(S | (NP | *The*	*The*	GEN(*hungry*)
4	(S | (NP | *The* | *hungry*	*The* | *hungry*	GEN(*cat*)
5	(S | (NP | *The* | *hungry* | *cat*	*The* | *hungry* | *cat*	REDUCE
6	(S | (NP *The hungry cat*)	*The* | *hungry* | *cat*	NT(VP)
7	(S | (NP *The hungry cat*) | (VP	*The* | *hungry* | *cat*	GEN(*meows*)
8	(S | (NP *The hungry cat*) | (VP *meows*	*The* | *hungry* | *cat* | *meows*	REDUCE
9	(S | (NP *The hungry cat*) | (VP *meows*)	*The* | *hungry* | *cat* | *meows*	GEN(.)
10	(S | (NP *The hungry cat*) | (VP *meows*) | .	*The* | *hungry* | *cat* | *meows* | .	REDUCE
11	(S (NP *The hungry cat*) (VP *meows*) .)	*The* | *hungry* | *cat* | *meows* | .	

提案手法

生成モデル

- 最大化: $p(X, Y; \Theta)$
- 単語列 $\boldsymbol{x}$ と構文木 $\boldsymbol{y}$ の結合確率

$$p(\boldsymbol{x}, \boldsymbol{y}) = \prod_{t=1}^{|\boldsymbol{a}(\boldsymbol{x},\boldsymbol{y})|} p(a_t|\boldsymbol{a}_{<t})$$

$$p(a_t|\boldsymbol{a}_{<t}) = \frac{\exp(\boldsymbol{r}_{a_t}^T \boldsymbol{u}_t + b_{a_t})}{\sum_{a' \in A_G(T_t, S_t, n_t)} \exp(\boldsymbol{r}_{a'}^T \boldsymbol{u}_t + b_{a'})}$$

$$\begin{cases} \boldsymbol{a}(x, y) : \text{単語列 } x \text{ と構文木 } y \text{ に対応する行動の列} \\ \boldsymbol{u}_t : \text{アルゴリズムの状態を表す埋め込み} \\ \boldsymbol{r}_a : \text{生成器の各行動の埋め込み（パラメータ）} \\ b_a : \text{生成器の各行動のバイアス（パラメータ）} \end{cases}$$

提案手法

生成モデル

- $\boldsymbol{u}_t$: アルゴリズムの状態を表す埋め込み

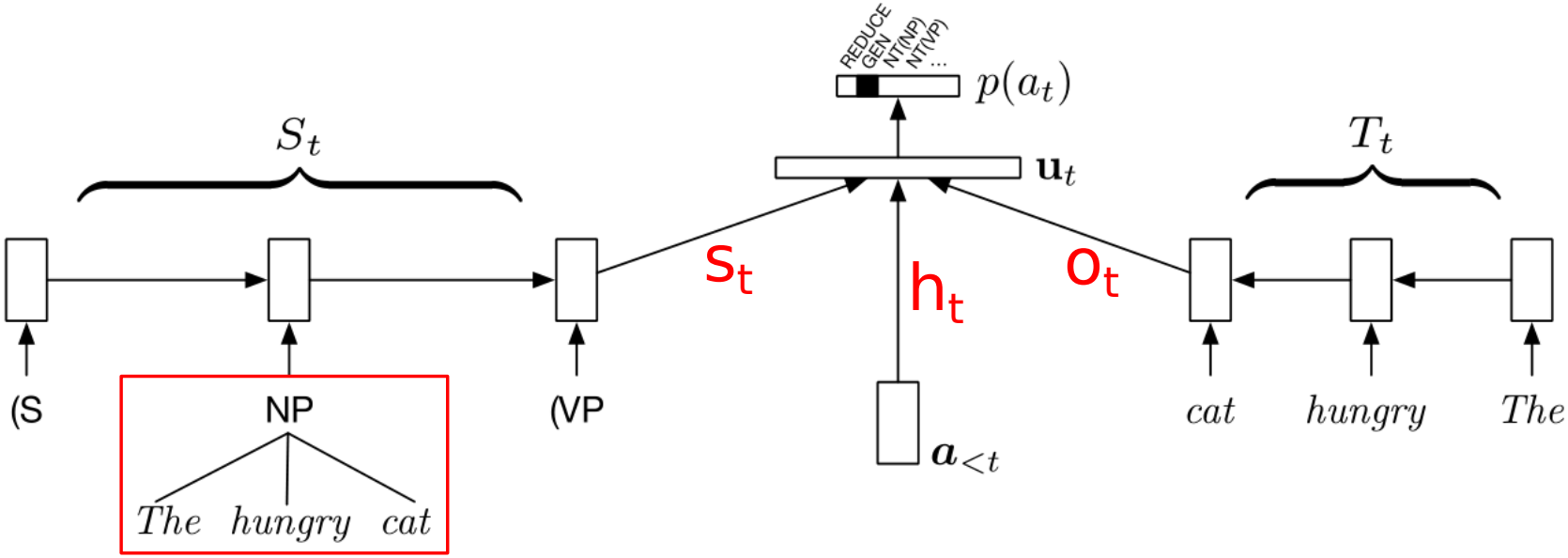
$$\boldsymbol{u}_t = \tanh(\boldsymbol{W}[\boldsymbol{o}_t; \boldsymbol{s}_t; \boldsymbol{h}_t] + \boldsymbol{c})$$

$$\begin{cases} \boldsymbol{o}_t : \text{出力バッファの状態を表す埋め込み} \\ \boldsymbol{s}_t : \text{スタックの状態を表す埋め込み} \\ \boldsymbol{h}_t : \text{行動履歴を表す埋め込み} \\ \boldsymbol{W}, \boldsymbol{c} : \text{パラメータ} \end{cases}$$

提案手法

生成モデル

- スタック・出力バッファ・行動履歴内の要素の埋め込みをエンコード

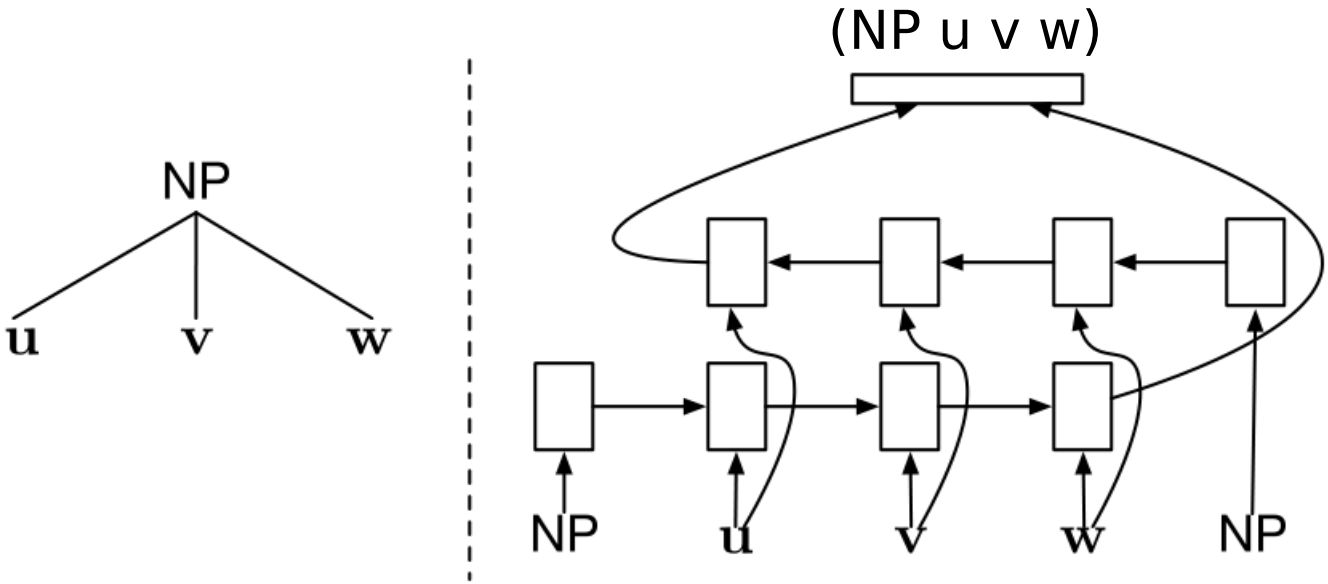


- スタック内の非終端記号の埋め込みは？

提案手法

生成モデル

- Syntactic Composition Function
  - REDUCE 時に要素の埋め込みから
    その非終端記号の埋め込みを生成



提案手法

識別モデル

- 最大化: $p(Y|X;\Theta)$
- 生成モデルの出力バッファ T を入力バッファ B に置き換え
- 単語列 $\boldsymbol{x}$ が与えられているため,
  先ほどの $p(\boldsymbol{x},\boldsymbol{y})$ を $p(\boldsymbol{y}|\boldsymbol{x})$ として学習

提案手法

生成モデルにおける重点サンプリング

- 提案分布 $q(\boldsymbol{y}|\boldsymbol{x})$：識別モデルを利用
- 重要度重み：$w(\boldsymbol{x}, \boldsymbol{y}) = p(\boldsymbol{x}, \boldsymbol{y})/q(\boldsymbol{y}|\boldsymbol{x})$
- $\boldsymbol{y}$ を q からサンプリングしモンテカルロ法で計算
- 言語モデル

$$
\begin{aligned}
p(\boldsymbol{x}) &= \sum_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})} p(\boldsymbol{x}, \boldsymbol{y}) \\
&= \sum_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})} q(\boldsymbol{y}|\boldsymbol{x}) w(\boldsymbol{x}, \boldsymbol{y}) \\
&= E_{q(\boldsymbol{y}|\boldsymbol{x})} w(\boldsymbol{x}, \boldsymbol{y})
\end{aligned}
$$

- 構文解析

$$
\hat{\boldsymbol{y}} = \mathrm{argmax}_{\boldsymbol{y} \sim q(\boldsymbol{y}|\boldsymbol{x})} p(\boldsymbol{x}, \boldsymbol{y})
$$

実験

実験設定

- データセット
  - Penn Treebank（英語）
  - Penn Chinese Treebank（中国語）
- タスク
  - 構文解析（生成・識別モデル）
  - 言語モデル（生成モデル）

実験

Penn Treebank での F 値

Table 2: Parsing results on PTB §23 (D=discriminative, G=generative, S=semisupervised).

Model	**type**	**F_1**
Henderson (2004)	D	89.4
Socher et al. (2013a)	D	90.4
Zhu et al. (2013)	D	90.4
Vinyals et al. (2015) – WSJ only	D	90.5
Petrov and Klein (2007)	G	90.1
Bod (2003)	G	90.7
Shindo et al. (2012) – single	G	91.1
Shindo et al. (2012) – ensemble	G	92.4
Zhu et al. (2013)	S	91.3
McClosky et al. (2006)	S	92.1
Vinyals et al. (2015) – single	S	92.5
Vinyals et al. (2015) – ensemble	S	92.8
Discriminative, $q(\boldsymbol{y} \mid \boldsymbol{x})$	D	89.8
Generative, $\hat{p}(\boldsymbol{y} \mid \boldsymbol{x})$	G	92.4

実験

Penn Chinese Treebank での F 値

Table 3: Parsing results on CTB 5.1.

Model	**type**	$\mathbf{F_1}$
Zhu et al. (2013)	D	82.6
Wang et al. (2015)	D	83.2
Huang and Harper (2009)	D	84.2
Charniak (2000)	G	80.8
Bikel (2004)	G	80.6
Petrov and Klein (2007)	G	83.3
Zhu et al. (2013)	S	85.6
Wang and Xue (2014)	S	86.3
Wang et al. (2015)	S	86.6
Discriminative, $q(\boldsymbol{y} \mid \boldsymbol{x})$	D	80.7
Generative, $\hat{p}(\boldsymbol{y} \mid \boldsymbol{x})$	G	82.7

- 識別モデルの方が生成モデルより性能低い
- 大きく構造化されていない条件付きの文脈は識別モデルにとって学習が難しい

実験

言語モデルの perplexity

Table 4: Language model perplexity results.

Model	**test ppl** (PTB)	**test ppl** (CTB)
IKN 5-gram	169.3	255.2
LSTM LM	113.4	207.3
RNNG	102.4	171.9

まとめ

- RNN による文法のモデルを提案
- 遷移ベースの構文解析・文生成アルゴリズムで学習
- 生成モデルと識別モデルの 2 種類
- State-of-the-art な手法と同等の性能

感想

- 構文解析はよく 知ら なかっ たので勉強になっ た
- 文法を NN でモデル化し ようという 発想が面白いと 思っ た
- スタッ クのエンコード が sequential の場合と の比較が知り たい

補足資料

提案手法

構文解析のアルゴリズム

$$\begin{cases} x : \text{終端記号（単語）の列（入力）} \\ y : \text{構文木（出力）} \\ S : \text{スタック} \\ B : \text{入力バッファ} \end{cases}$$

- スタックの要素：終端記号，open または closed な非終端記号
- 入力バッファの要素：終端記号
- 初期状態

$$\begin{cases} S = \emptyset \\ B = [T_1, \ldots, T_n] \end{cases}$$

提案手法

構文解析のアルゴリズム

Stack$_t$	**Buffer**$_t$	**Open NTs**$_t$	**Action**	**Stack**$_{t+1}$	**Buffer**$_{t+1}$	**Open NTs**$_{t+1}$
S	B	n	NT(X)	$S \mid (\mathrm{X}$	B	$n+1$
S	$x \mid B$	n	SHIFT	$S \mid x$	B	n
$S \mid (\mathrm{X} \mid \tau_1 \mid \ldots \mid \tau_\ell$	B	n	REDUCE	$S \mid (\mathrm{X}\ \tau_1\ \ldots\ \tau_\ell)$	B	$n-1$

- 遷移の制約
  - n: スタック内の open な非終端記号の数

遷移	制約
NT(X)	$B \neq \emptyset \wedge n < 100$
SHIFT	$B \neq \emptyset \wedge n \geq 1$
REDUCE	スタック内の一番上の要素が open な非終端記号でない $\wedge\, (n \geq 2 \vee B = \emptyset)$

提案手法

文生成のアルゴリズム

$$\begin{cases} x: \text{終端記号（単語）の列（出力）} \\ y: \text{構文木（出力）} \\ S: \text{スタック} \\ \textbf{T: 出力バッファ} \end{cases}$$

- スタックの要素：終端記号，open または closed な非終端記号
- 出力バッファの要素：終端記号
- 初期状態

$$\begin{cases} S = \emptyset \\ T = \emptyset \end{cases}$$

提案手法

文生成のアルゴリズム

Stack$_t$	**Terms**$_t$	**Open NTs**$_t$	**Action**	**Stack**$_{t+1}$	**Terms**$_{t+1}$	**Open NTs**$_{t+1}$
S	T	n	NT(X)	$S \mid (\mathrm{X}$	T	$n+1$
S	T	n	GEN(x)	$S \mid x$	$T \mid x$	n
$S \mid (\mathrm{X} \mid \tau_1 \mid \ldots \mid \tau_\ell$	T	n	REDUCE	$S \mid (\mathrm{X}\ \tau_1\ \ldots\ \tau_\ell)$	T	$n-1$

- 遷移の制約
  - n: スタック内の open な非終端記号の数

遷移	制約
GEN(X)	$n \geq 1$
REDUCE	スタック内の一番上の要素が open な非終端記号でない $\wedge\ n \geq 1$

提案手法

生成モデルにおける単語の推定

- Class-factored softmax
  - $|\Sigma|$：語彙数
  - Brown et al., 2016 の手法により語彙を $\sqrt{|\Sigma|}$ 個のクラスタに分割
  - 単語の推定の計算量：$O(\sqrt{|\Sigma|})$

$$p(x) = p(x|c)p(c)$$