Recurrent Neural Network Grammars

王希豪

北京大学中国语言文学系

wangxihao@pku.edu.cn

2022 年 5 月 12 日

Overview

Motivation

*Despite these impressive results, sequential models are a **priori inappropriate** models of natural language, since relationships among words are largely organized in terms of latent nested structures rather than sequential surface order[Chomsky, 1957].*

尽管序列化模型在各种任务中取得了令人印象深刻的效果，但对于自然语言来说，序列化模型是一类先验不恰当的模型。因为自然语言中的词语是根据内在的嵌套结构组织的，而不仅仅是按照表面的序列化顺序组织的。

Chomsky Hierarchy

乔姆斯基层级

类别	文法	自动机
Type 3	正则文法	有限状态自动机 (FSA)
Type 2	上下文无关文法 (CFG)	非确定性下推自动机 (NPDA)
Type 1	上下文相关文法 (CSG)	线性有限自动机 (LBA)
Type 0	无限制文法	图灵机 (Turing Machine)

Chomsky Hierarchy

乔姆斯基层级

类别	文法	自动机
Type 3	正则文法	有限状态自动机 (FSA)
Type 2	上下文无关文法 (CFG)	非确定性下推自动机 (NPDA)
Type 1	上下文相关文法 (CSG)	线性有限自动机 (LBA)
Type 0	无限制文法	图灵机 (Turing Machine)

Type 0 语言是否就是自然语言？

Chomsky Hierarchy

乔姆斯基层级

类别	文法	自动机
Type 3	正则文法	有限状态自动机 (FSA)
Type 2	上下文无关文法 (CFG)	非确定性下推自动机 (NPDA)
Type 1	上下文相关文法 (CSG)	线性有限自动机 (LBA)
Type 0	无限制文法	图灵机 (Turing Machine)

Type 0 语言是否就是自然语言？

两种语法分析任务

- 成分分析 (Constituent Parsing)
- 依存分析 (Dependency Parsing)

Chomsky Hierarchy

乔姆斯基层级

类别	文法	自动机
Type 3	正则文法	有限状态自动机 (FSA)
Type 2	上下文无关文法 (CFG)	非确定性下推自动机 (NPDA)
Type 1	上下文相关文法 (CSG)	线性有限自动机 (LBA)
Type 0	无限制文法	图灵机 (Turing Machine)

Type 0 语言是否就是自然语言？

两种语法分析任务

- 成分分析 (Constituent Parsing)
- 依存分析 (Dependency Parsing)

Chomsky Hierarchy

乔姆斯基层级

类别	文法	自动机
Type 3	正则文法	有限状态自动机 (FSA)
Type 2	上下文无关文法 (CFG)	非确定性下推自动机 (NPDA)
Type 1	上下文相关文法 (CSG)	线性有限自动机 (LBA)
Type 0	无限制文法	图灵机 (Turing Machine)

Type 0 语言是否就是自然语言？

两种语法分析任务

- 成分分析 (Constituent Parsing)
- 依存分析 (Dependency Parsing)

理由

- 语言生成中最核心的两个机制是合并 (Merge) 与递归 (Recursion)。

Motivation

Despite these impressive results, sequential models are a ***priori inappropriate*** *models of natural language, since relationships among words are largely organized in terms of latent nested structures rather than sequential surface order[Chomsky, 1957].*

尽管序列化模型在各种任务中取得了令人印象深刻的效果，但对于自然语言来说，序列化模型是一类先验不恰当的模型。因为自然语言中的词语是根据内在的嵌套结构组织的，而不仅仅是按照表面的序列化顺序组织的。

Dyer et al. [2016] **提出了一类新的生成概率模型，显性地对短语之间的层次结构进行了建模。他将这类模型称为循环神经网络语法** (Recurrent Neural Network Grammars, RNNGs)。

Introduction

Recurrent Neural Network Grammars (RNNGs) is a new generative probabilistic model of sentences that explicitly models nested, hierarchical relationships among words and phrases. RNNGs operate via a recursive syntactic process reminiscent of probabilistic context-free grammar generation, but decisions are parameterized using RNNs that condition on the entire syntactic derivation history, greatly relaxing context-free independence assumptions.

RNNGs 是一种新的生成概率模型，它显性地对句子中词和短语嵌套的层次关系进行了建模。*RNNGs* 生成句子的过程和 *PCFG* 类似，都是通过递归的语法操作进行。但是，*RNNG* 在进行决策的时候，使用了循环神经网络对整个句法派生历史进行了编码，大大地放松了上下文无关的独立性假设。

Introduction

Recurrent Neural Network Grammars (RNNGs) is a new generative probabilistic model of sentences that explicitly models nested, hierarchical relationships among words and phrases. RNNGs operate via a recursive syntactic process reminiscent of probabilistic context-free grammar generation, but decisions are parameterized using RNNs that condition on the entire syntactic derivation history, greatly relaxing context-free independence assumptions.

RNNGs 是一种新的生成概率模型，它显性地对句子中词和短语嵌套的层次关系进行了建模。*RNNGs* 生成句子的过程和 *PCFG* 类似，都是通过递归的语法操作进行。但是，*RNNG* 在进行决策的时候，使用了循环神经网络对整个句法派生历史进行了编码，大大地放松了上下文无关的独立性假设。

RNNGs 是一种概率语言模型，和基于马尔科夫链的概率语言模型不同的是，RNNGs 是对句子的短语结构进行了建模。因此，它可以 parse 输入的句子片段，并对下一个词语进行预测。

Overview

A **top-down** variant of **transition-based** parsing

Parser Transitions

- 两个数据结构
  - 栈 (Stack)
  - 输入缓存区 (Input Buffer)

A **top-down** variant of **transition-based** parsing

Parser Transitions

- 两个数据结构
  - 栈 (Stack)
  - 输入缓存区 (Input Buffer)
- 三种操作
  - nt(X): 将一个开放的非终结符 X 压入到栈中
  - shift: 将输入缓存区顶部的终结符压入到栈中
  - reduce: 将栈中的元素依次弹出，直到发现了一个开放的非终结符 X，将其转化为非开放的终结符再次压入到栈中。意味着有一个结点的所有子节点已经填充完。

A **top-down** variant of **transition-based** parsing

Parser Transitions

- 两个数据结构
  - 栈 (Stack)
  - 输入缓存区 (Input Buffer)
- 三种操作
  - nt(X): 将一个开放的非终结符 X 压入到栈中
  - shift: 将输入缓存区顶部的终结符压入到栈中
  - reduce: 将栈中的元素依次弹出，直到发现了一个开放的非终结符 X，将其转化为非开放的终结符再次压入到栈中。意味着有一个结点的所有子节点已经填充完。

$\textbf{Stack}_t$	$\textbf{Buffer}_t$	$\textbf{Open NTs}_t$	Action	$\textbf{Stack}_{t+1}$	$\textbf{Buffer}_{t+1}$	$\textbf{Open NTs}_{t+1}$
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	$n-1$

图: 三种操作对栈、输入缓存区和开放非终结符数量的影响

A **top-down** variant of **transition-based** parsing

Parser Transitions

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*) .)		

图: Parsing 示例

A **top-down** variant of **transition-based** parsing

Parser Transitions

- 两个数据结构
  - 栈 (Stack): 简记为 S
  - 输入缓存区 (Input Buffer): 简记为 B
- 三种操作
  - nt(X): 将一个开放的非终结符 X 压入到栈中
  - shift: 将输入缓存区顶部的终结符压入到栈中
  - reduce: 将栈中的元素依次弹出，直到发现了一个开放的非终结符 X，将其转化为非开放的终结符再次压入到栈中。意味着有一个结点的所有子节点已经填充完。
- 限制条件
  - 只有当 B 非空且 $n<100$ 时，才会执行 nt(X) 操作
  - 只有当 B 非空且 $n\geq 1$ 时才会执行 shift 操作
  - 只有当 S 的顶端不是开放非终结符时才会执行 reduce 操作
  - 只有当 $n\geq 2$ 或 B 为空时才会执行 reduce 操作

A **top-down** variant of **transition-based** parsing

Parser Transitions

- 两个数据结构
  - 栈 (Stack): 简记为 S
  - 输入缓存区 (Input Buffer): 简记为 B
- 三种操作
  - nt(X): 将一个开放的非终结符 X 压入到栈中
  - shift: 将输入缓存区顶部的终结符压入到栈中
  - reduce: 将栈中的元素依次弹出，直到发现了一个开放的非终结符 X，将其转化为非开放的终结符再次压入到栈中。意味着有一个结点的所有子节点已经填充完。
- 限制条件
  - 只有当 B 非空且 $n<100$ 时，才会执行 nt(X) 操作
  - 只有当 B 非空且 $n\geq 1$ 时才会执行 shift 操作
  - 只有当 S 的顶端不是开放非终结符时才会执行 reduce 操作
  - 只有当 $n\geq 2$ 或 B 为空时才会执行 reduce 操作
- 某时刻可行的 parser transitions 的集合记为 $\mathcal{A}_D(B, S, n)$

A **top-down** variant of **transition-based** parsing

Generator Transitions

- 两个数据结构
  - 栈 (Stack): 简记为 S
  - 输出缓存区 (Output Buffer): 简记为 T

A **top-down** variant of **transition-based** parsing

Generator Transitions

- 两个数据结构
  - 栈 (Stack): 简记为 S
  - 输出缓存区 (Output Buffer): 简记为 T
- 三种操作
  - nt(X): 将一个开放的非终结符 X 压入到栈中
  - gen(x): 生成终结符 x 并将其分别压入到栈和输出缓存区中
  - reduce: 将栈中的元素依次弹出，直到发现了一个开放的非终结符 X，将其转化为非开放的终结符再次压入到栈中。意味着有一个结点的所有子节点已经填充完。

A **top-down** variant of **transition-based** parsing

Generator Transitions

- 两个数据结构
  - 栈 (Stack): 简记为 S
  - 输出缓存区 (Output Buffer): 简记为 T
- 三种操作
  - nt(X): 将一个开放的非终结符 X 压入到栈中
  - gen(x): 生成终结符 x 并将其分别压入到栈和输出缓存区中
  - reduce: 将栈中的元素依次弹出，直到发现了一个开放的非终结符 X，将其转化为非开放的终结符再次压入到栈中。意味着有一个结点的所有子节点已经填充完。

$\textbf{Stack}_t$	$\textbf{Terms}_t$	$\textbf{Open NTs}_t$	Action	$\textbf{Stack}_{t+1}$	$\textbf{Terms}_{t+1}$	$\textbf{Open NTs}_{t+1}$
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	$n-1$

图: 三种操作对栈、输出缓存区和开放非终结符数量的影响

A **top-down** variant of **transition-based** parsing

Generator Transitions

	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* | .	

图: Generation 示例

A **top-down** variant of **transition-based** parsing

Generator Transitions

- 两个数据结构
  - 栈 (Stack): 简记为 S
  - 输出缓存区 (Output Buffer): 简记为 T
- 三种操作
  - nt(X): 将一个开放的非终结符 X 压入到栈中
  - gen(x): 生成终结符 x 并将其分别压入到栈和输出缓存区中
  - reduce: 将栈中的元素依次弹出，直到发现了一个开放的非终结符 X，将其转化为非开放的终结符再次压入到栈中。意味着有一个结点的所有子节点已经填充完。
- 限制条件
  - 只有当 $n \geq 1$ 时才会执行 gen(x) 操作
  - 只有当 S 的顶端不是开放非终结符且 $n \geq 1$ 时才会执行 reduce 操作

A **top-down** variant of **transition-based** parsing

Generator Transitions

- 两个数据结构
  - 栈 (Stack): 简记为 S
  - 输出缓存区 (Output Buffer): 简记为 T
- 三种操作
  - nt(X): 将一个开放的非终结符 X 压入到栈中
  - gen(x): 生成终结符 x 并将其分别压入到栈和输出缓存区中
  - reduce: 将栈中的元素依次弹出，直到发现了一个开放的非终结符 X，将其转化为非开放的终结符再次压入到栈中。意味着有一个结点的所有子节点已经填充完。
- 限制条件
  - 只有当 $n \geq 1$ 时才会执行 gen(x) 操作
  - 只有当 S 的顶端不是开放非终结符且 $n \geq 1$ 时才会执行 reduce 操作
- 某时刻可行的 generator transitions 的集合记为 $\mathcal{A}_G(T, S, n)$

Generative Model

数学原理

- 记词序列为 $\mathbf{x}$，句法树为 $\mathbf{y}$

Generative Model

数学原理

- 记词序列为 $\mathbf{x}$，句法树为 $\mathbf{y}$
- RNNGs 利用 generator transitions 定义了两者之间的联合概率分布

$$
\begin{aligned}
p(\mathbf{x}, \mathbf{y}) &= \prod_{t=1}^{|\mathbf{a}(\mathbf{x},\mathbf{y})|} p(a_t | \mathbf{a}_{<t}) \\
&= \prod_{t=1}^{|\mathbf{a}(\mathbf{x},\mathbf{y})|} \frac{\exp(\mathbf{r}_{a_t}^\top \mathbf{u}_t + b_{a_t})}{\sum_{a' \in \mathcal{A}_G(T_t, S_t, n_t)} \exp(\mathbf{r}_{a'}^\top \mathbf{u}_t + b_{a'})} \\
\mathbf{u}_t &= \tanh(\mathbf{W}[\mathbf{o}_t; \mathbf{s}_t; \mathbf{h}_t] + \mathbf{c})
\end{aligned}
$$

其中，$\mathbf{o}_t$ 是对输出缓存区的编码，$\mathbf{s}_t$ 是对栈的编码，$\mathbf{h}_t$ 是对操作历史的编码，$\mathbf{W}$ 和 $\mathbf{c}$ 是参数

Generative Model

数学原理

- 记词序列为 $\mathbf{x}$，句法树为 $\mathbf{y}$
- RNNGs 利用 generator transitions 定义了两者之间的联合概率分布

框架示例

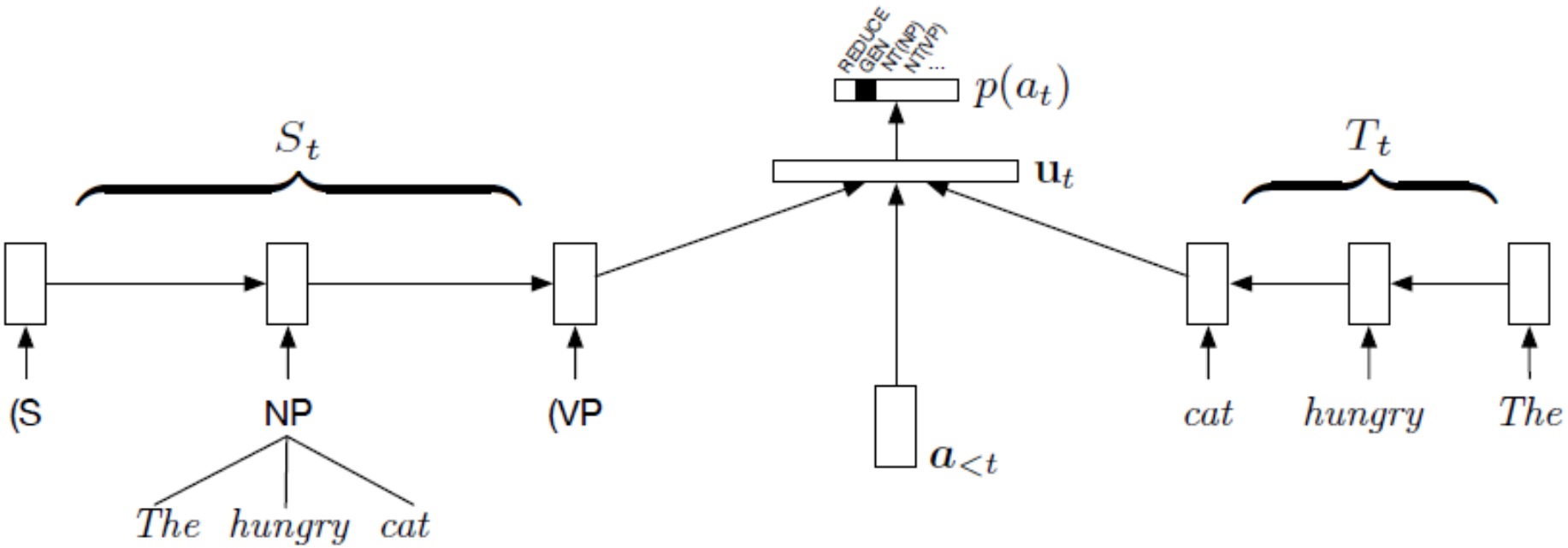

图: RNNGs 模型框架示例

Generative Model

数学原理

- 记词序列为 $\mathbf{x}$，句法树为 $\mathbf{y}$
- RNNGs 利用 generator transitions 定义了两者之间的联合概率分布

编码

- 输出缓冲区和操作历史都是由有限符号组成的序列，直接使用 RNN 进行编码
- 对于栈中的数据，利用基于双向 LSTM 的组合函数进行编码

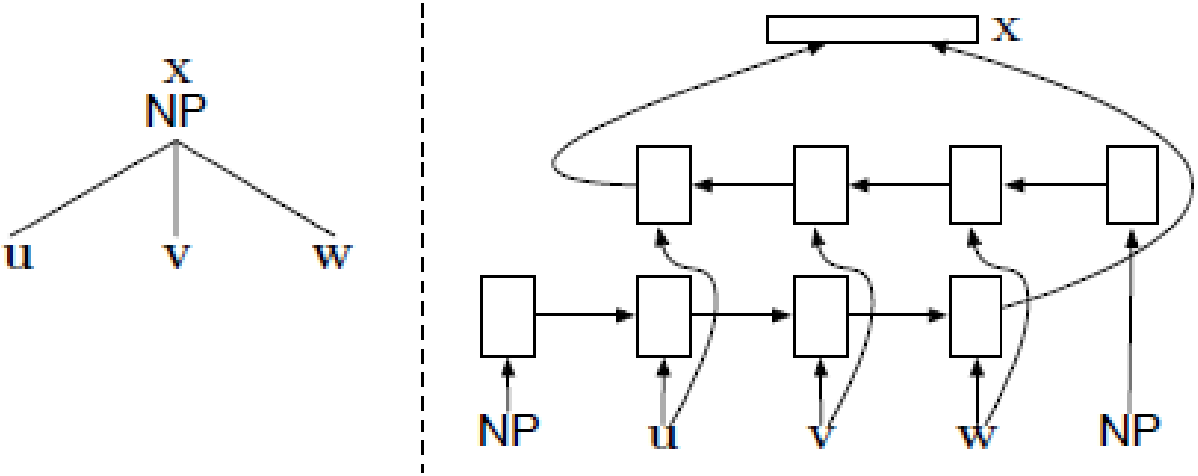

图: 组合函数

Inference

Parsing

$$\grave{}\,;\ \mathsf{K}\quad \not{p}(\mathbf{x},\mathbf{y})$$
$$\mathbf{y}\in\mathcal{Y}(x)$$

Language Model

$$p(\mathbf{x}) = \mathsf{X}\qquad p(\mathbf{x},\mathbf{y}^{0})$$
$$\mathbf{y}^{0}\in\mathcal{Y}(x)$$

Inference

Parsing

$$\grave{\ }\,;\ \mathsf{K}\quad \not{p}(\mathbf{x},\mathbf{y})$$
$$\mathbf{y}\in\mathcal{Y}(x)$$

Language Model

$$p(\mathbf{x}) = \sum_{\mathbf{y}^0\in\mathcal{Y}(x)} p(\mathbf{x},\mathbf{y}^0)$$

Importance Sampling

预先设定易于采样的概率分布$\mathrm{q}(\mathbf{y}|\mathbf{x})$

令 $w(\mathbf{x},\mathbf{y}) = p(\mathbf{x},\mathbf{y})/q(\mathbf{y}|\mathbf{x})$

则有 $p(\mathbf{x}) = \sum_{y\in\mathcal{Y}(x)} q(\mathbf{y}|\mathbf{x})w(\mathbf{x},\mathbf{y}) = \mathsf{E}_{q(\mathbf{y}|\mathbf{x})}w(\mathbf{x},\mathbf{y})$

Inference

Parsing

$$\arg\max_{\mathbf{y}\in\mathcal{Y}(x)} p(\mathbf{x},\mathbf{y})$$

Language Model

$$p(\mathbf{x}) = \sum_{\mathbf{y}^0\in\mathcal{Y}(x)} p(\mathbf{x},\mathbf{y}^0)$$

Importance Sampling

预先设定易于采样的概率分布$q(\mathbf{y}|\mathbf{x})$

利用蒙特卡洛算法进行采样 N 次

$$\mathbf{y}^{(i)} \sim q(\mathbf{y}|\mathbf{x}), i \in \{1,2,\cdots,N\}$$

$$\mathbb{E}_{q(\mathbf{y}|\mathbf{x})} w(\mathbf{x},\mathbf{y}) \overset{MC}{\approx} \frac{1}{N}\sum_{i=1}^{N} w(\mathbf{x},\mathbf{y}^{(i)})$$

Experiments

Model	type	F_1
Vinyals et al. (2015)* – WSJ only	D	88.3
Henderson (2004)	D	89.4
Socher et al. (2013a)	D	90.4
Zhu et al. (2013)	D	90.4
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	92.1
Discriminative, $q(\boldsymbol{y} \mid \boldsymbol{x})$	D	89.8
Generative, $\hat{p}(\boldsymbol{y} \mid \boldsymbol{x})$	G	92.4

图: PTB 的 parsing 结果

Experiments

Model	type	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, $q(y \mid x)$	D	80.7
Generative, $\hat{p}(y \mid x)$	G	82.7

图: CTB 的 parsing 结果

Experiments

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

图: 语言模型的困惑度

Overview

Composition is Key

框架示例

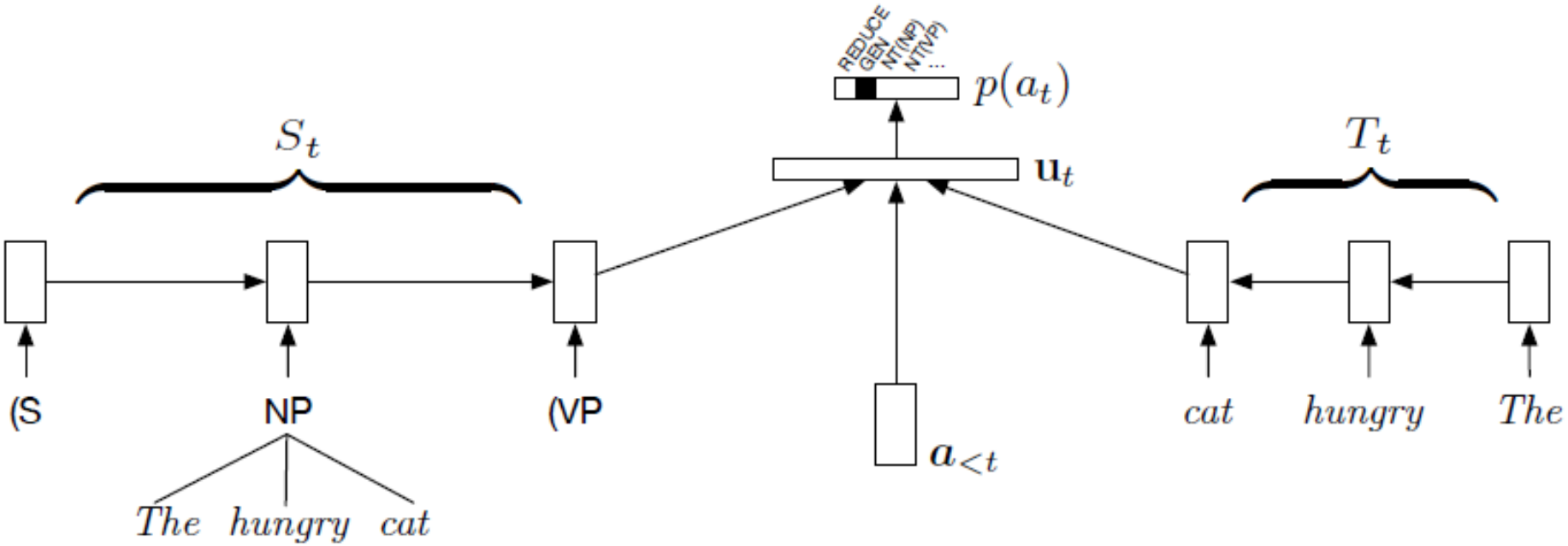

图: RNNGs 模型框架示例

Composition is Key

Ablated RNNGs 实验结果

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

图: Ablated RNNGs 在短语结构预测任务上的表现

Composition is Key

Ablated RNNGs 实验结果

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

图: Ablated RNNGs 在依存结构预测任务上的表现

Composition is Key

Ablated RNNGs 实验结果

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**

图: Ablated RNNGs 作为语言模型时的表现

Composition is Key

Ablated RNNGs 实验结果

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**

图: Ablated RNNGs 作为语言模型时的表现

结论

- 组合函数是影响 RNNGs 模型效果的关键因素

Gated Attention RNNG

语言学假设

- 短语的表示主要由头词来决定 [Chomsky, 1993, Collins, 1997]
- 短语的类别是否可以由其内在结构决定 [Chomsky et al., 1968]

Gated Attention RNNG

语言学假设

- 短语的表示主要由头词来决定 [Chomsky, 1993, Collins, 1997]
- 短语的类别是否可以由其内在结构决定 [Chomsky et al., 1968]

RNNGs 模型的组合函数

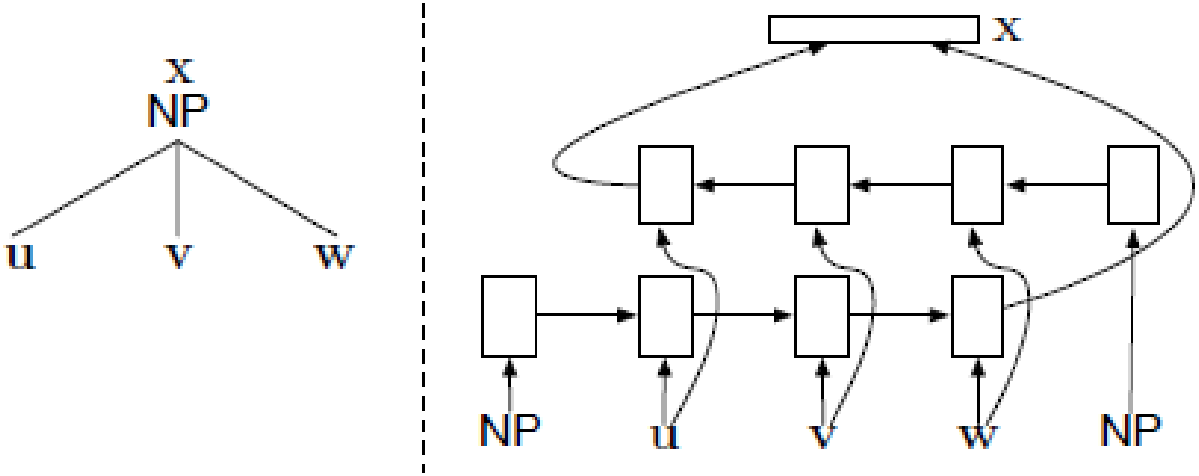

图: RNNGs 的组合函数

Gated Attention RNNG

语言学假设

- 短语的表示主要由头词来决定 [Chomsky, 1993, Collins, 1997]
- 短语的类别是否可以由其内在结构决定 [Chomsky et al., 1968]

GA-RNNGs 模型的组合函数

- $\mathbf{a} = \mathrm{softmax}([\mathbf{c}_1 \mathbf{c}_2 \cdots]^\top \mathbf{V}[\mathbf{u}; \mathbf{o}_{nt}])$
- 记 $\mathbf{m} = [\mathbf{c}_1; \mathbf{c}_2; \cdots]\mathbf{a}$
- 令 $\mathbf{g} = \sigma(\mathbf{W}_1 \mathbf{t}_{nt} + \mathbf{W}_2 \mathbf{m} + \mathbf{b})$
- 得到 $\mathbf{c} = \mathbf{g} \odot \mathbf{t}_{nt} + (\mathbf{1} - \mathbf{g}) \odot \mathbf{m}$

Headness

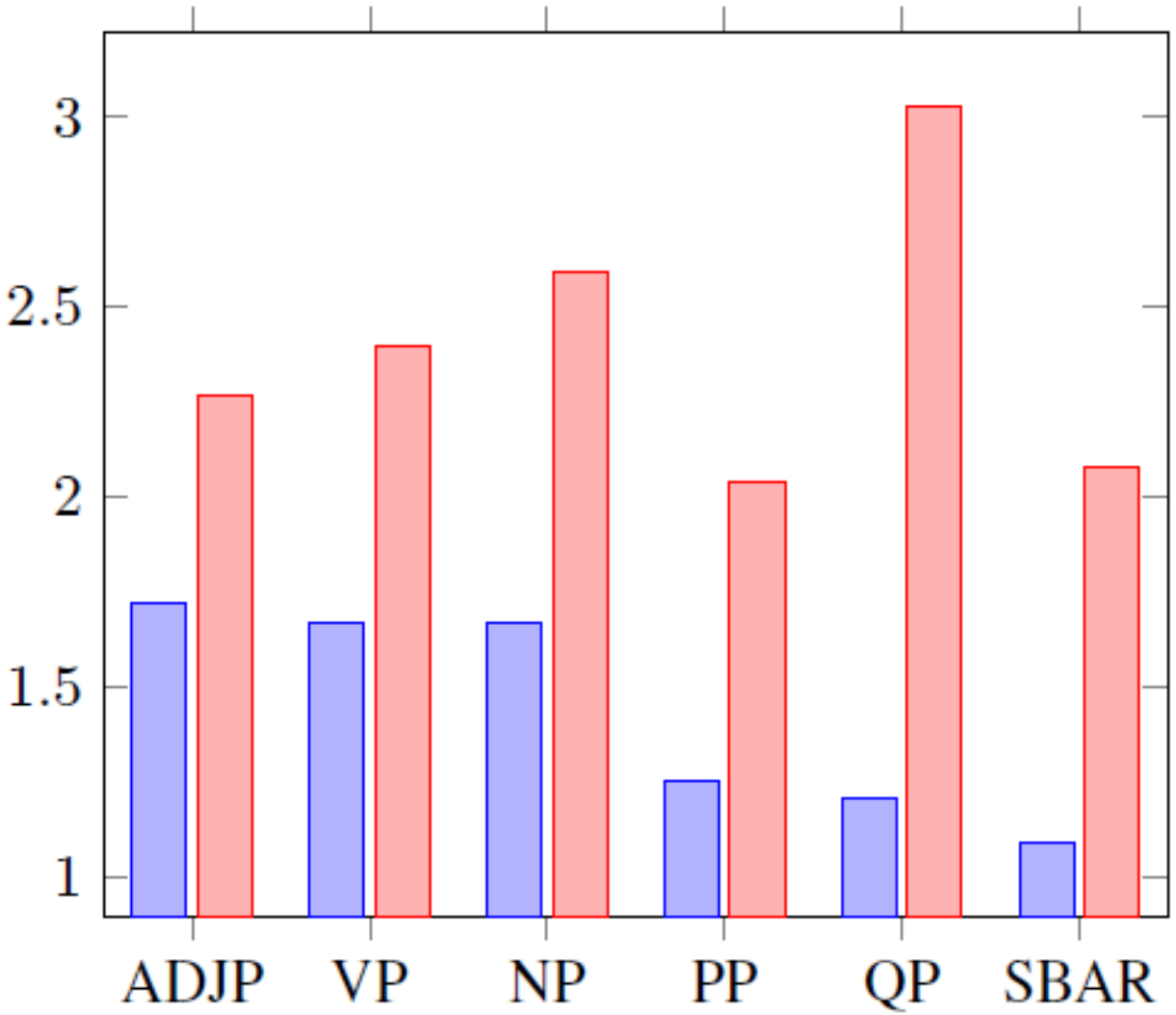

图: 各类短语注意力向量的平均困惑度

Headness

	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)

图: 短语注意力向量实例

The Role of Nonterminal Labels

标签引入的信息很少

- 基于不带标签数据训练的模型 U-GA-RNNG，parsing 的表现为 93.7
- 基于带标签数据训练的模型 GA-RNNG，parsing 的表现为 94.2

The Role of Nonterminal Labels

标签引入的信息很少

- 基于不带标签数据训练的模型 U-GA-RNNG，parsing 的表现为 93.7
- 基于带标签数据训练的模型 GA-RNNG，parsing 的表现为 94.2

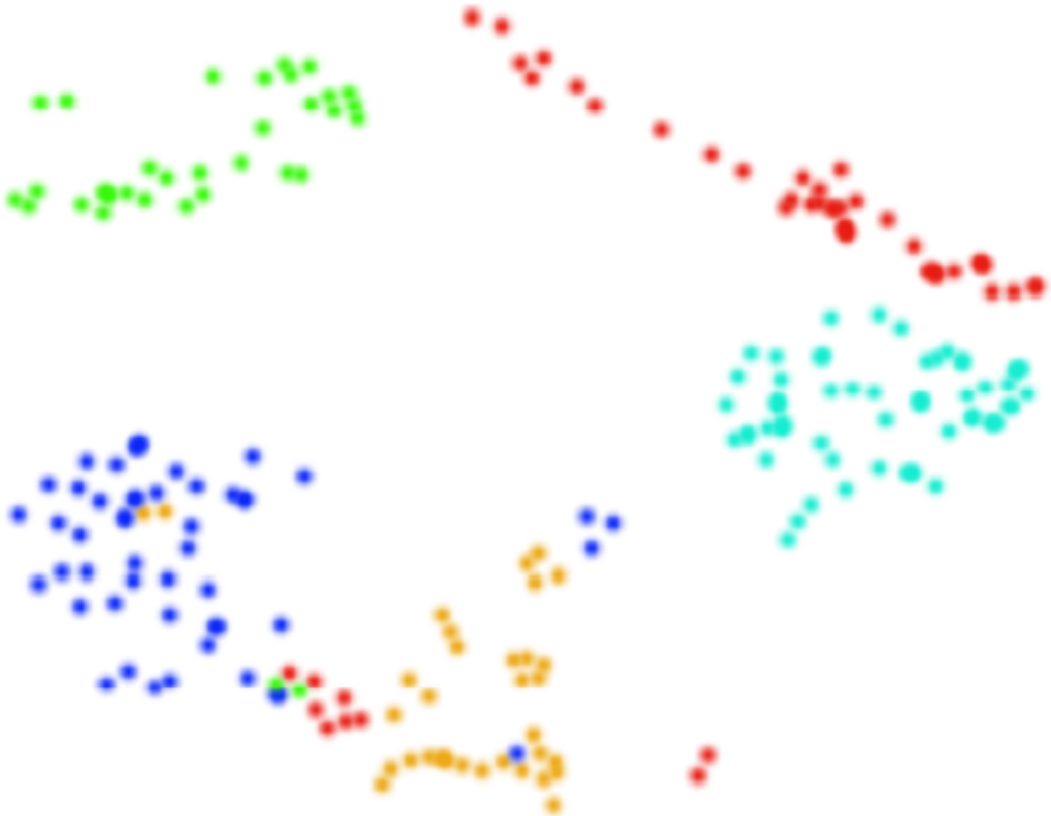

图: t-SNE 算法对 U-GA-RNNG 模型计算的短语向量进行聚类的结果。深蓝色的点是 VPs，红点是 SBARs，黄点是 PPs，浅蓝色的点是 NPs，绿点是 Ss。

Overview

Conclusions

总结

- RNNGs 模型成功地将句法结构和神经网络模型进行了结合
- 通过实验，可以发现 RNNGs 模型成功的关键在于对合并 (Merge) 操作的建模
- 根据 GA-RNNGs 的实验，我们可以发现模型学到了和语言学中 Head 概念类似的信息，而对 VPs 中异常的解释还需要进一步的研究和分析
- 从可视化结果可以证明，短语的类型一定程度上由其内部结构决定

Conclusions

总结

- RNNGs 模型成功地将句法结构和神经网络模型进行了结合
- 通过实验，可以发现 RNNGs 模型成功的关键在于对合并 (Merge) 操作的建模
- 根据 GA-RNNGs 的实验，我们可以发现模型学到了和语言学中 Head 概念类似的信息，而对 VPs 中异常的解释还需要进一步的研究和分析
- 从可视化结果可以证明，短语的类型一定程度上由其内部结构决定

Future works

- LSTMs Can Learn Syntax-Sensitive Dependencies Well, But Modeling Structure Makes Them Better[Kuncoro et al., 2018]
- Unsupervised Recurrent Neural Network Grammars[Kim et al., 2019]
- Transformer Grammars: Augmenting Transformer Language Models with Syntactic Inductive Biases at Scale[Sartran et al., 2022]

参考文献 I

N. Chomsky. Syntactic structures. 1957.

N. Chomsky. A minimalist program for linguistic theory. The view from Building 20: Essays in linguistics in honor of Sylvain Bromberger, 1993.

N. Chomsky et al. Remarks on nominalization. Linguistics Club, Indiana University, 1968.

M. Collins. Three generative, lexicalised models for statistical parsing. arXiv preprint cmp-lg/9706022, 1997.

C. Dyer, A. Kuncoro, M. Ballesteros, and N. A. Smith. Recurrent neural network grammars. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 199–209, San Diego, California, June 2016. Association for Computational Linguistics. doi:
R y X R 3 e 8 j f p R f L UR L @ R y k 9
? i i T b , f f + H M i ? Q H Q ; v X Q ` ; f L R e @ R y k 9

参考文献 II

Y. Kim, A. M. Rush, L. Yu, A. Kuncoro, C. Dyer, and G. Melis. Unsupervised recurrent neural network grammars. arXiv preprint arXiv:1904.03746, 2019.

A. Kuncoro, C. Dyer, J. Hale, D. Yogatama, S. Clark, and P. Blunsom. Lstms can learn syntax-sensitive dependencies well, but modeling structure makes them better. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1426–1436, 2018.

L. Sartran, S. Barrett, A. Kuncoro, M. Stanojević, P. Blunsom, and C. Dyer. Transformer grammars: Augmenting transformer language models with syntactic inductive biases at scale. arXiv preprint arXiv:2203.00633, 2022.