

一种针对句法树的混合神经网络模型

霍欢^{1,2}, 张薇¹, 刘亮¹, 李洋¹

(1. 上海理工大学 光电信息与计算机工程学院, 上海 200093;
2. 复旦大学 上海市数据科学重点实验室, 上海 201203)

摘要: 在多数神经网络模型仍然将目光放在顺序结构上时, 近期出现的两种基于句法树的模型 TreeLSTMs 和 TBCNNs 由于加入了结构信息而在多个自然语言处理任务上表现出色。考虑到 TreeLSTMs 因计算空间关联性使其训练效率不高, 为了改进这一缺点, 本文提出一种针对句法树的混合神经网络模型, 借助 TBCNNs 的树卷积和池化方法实现了类似 TreeLSTMs 的计算, 故将此模型命名为 Quasi-TreeLSTMs。本文在依存树和支持树上分别构建了模型的两种版本 *Dependency Quasi-TreeLSTMs* 和 *Constituency Quasi-TreeLSTMs*, 实验结果表明, 在情感分类和语义相似性两类任务上 Quasi-TreeLSTMs 的表现优异。

关键词: 句法树; TreeLSTMs; TBCNNs; 并行性; 混合模型

中图分类号: TP391 **文献标识码:** A

A Hybrid Neural Network Model on Syntax Tree Structures

HUO Huan^{1,2}, ZHANG Wei¹, LIU Liang¹, LI Yang¹

(1.School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, Shanghai 200093, P.R. China;

2.Shanghai Key Laboratory of Data Science, Fudan University, Shanghai 201203, P.R. China)

Abstract: When most of the neural network models still focus on the sequential texts, two recently-proposed tree-based models, TreeLSTMs and TBCNNs, perform well on multiple natural language processing tasks by the means of structural information. Since the former faces a serious problem of inefficient training due to the structure-related computations, this paper proposes a hybrid neural network model Quasi-TreeLSTMs, which is based on the tree convolution and pooling method of TBCNNs to mimic the TreeLSTMs' operations. The model has two different variants *Dependency Quasi-TreeLSTMs* and *Constituency Quasi-TreeLSTMs*, in accordance with two kinds of syntax trees. The experimental results show that the performance of Quasi-TreeLSTMs is excellent in both sentiment classification and semantic similarity tasks.

Key words: Syntax tree; TreeLSTMs; TBCNNs; parallel computing; hybrid model

1 引言

文本处理模型大致可以归为三类: BOW (Bag-of-Words) 模型、序列化模型和基于句法树的模型。相对于 BOW 模型^[1,2]词与词间的独立性假设, 序列化模型^[3,4]考虑了词序信息, 并因其突出性能被广泛使用。但前两种模型都忽略了文本自身存在的句法结构, 而句法结构对获取文本语义特征相当重要。因此, TreeLSTMs 模型^[5]是一种针对句法树的 LSTMs 模型, 该模型将顺序处理的 LSTM cells 按句法树递归排布, 使原本 $o(n)$ 的操作变成 $o(\log(n))$, 缩短了反向传播的路径, 在一定程度上

缓解了梯度消失的问题, 使模型能够更准确地学习长序列的空间关联性。其中句法树是将句子借助于树形图来说明句中词与词、词组与词组之间的句法、语义和逻辑关系。目前树形结构分为两种: 支持树 (constituency tree) 和依存树 (Dependency Tree), 分别如图 1 (a) 和 (b) 所示。其中, (a) 中加粗箭头表示组合, (b) 中加粗箭头表示卷积操作。

但 TreeLSTMs 的缺点也显而易见: 在图 1 (a) 中, 为了计算父节点 O_1 的隐藏状态 h_1 和细胞状态 c_1 , 首先要获得它两个子节点 O_2 和 O_3 的隐藏状态和细胞状态, 然后再加以组合。这种对空间关联性的计算完全限制了 TreeLSTMs 的并行能力, 在需要训练

收稿日期: 2017-05-31

定稿日期: 2017-07-25

基金项目: 国家自然科学基金 (61003031), 上海重点科技攻关项目 (14511107902), 上海市工程中心建设项目 (GCZX14014), 上海市一流学科建设项目 (XTKX2012), 上海市数据科学重点实验室开放课题资助课题 (201609060003), 沪江基金研究基地专项 (C14001)。

作者简介: 霍欢 (出生年 1979), 女, 博士, 副教授, 主要研究方向: 数据管理、数据分析及数据挖掘, CCF 会员; 张薇 (出生年 1993), 女, 硕士, 主要研究方向: 数据清洗与数据质量; 刘亮 (出生年 1991), 男, 硕士, 研究方向: 机器学习; 李洋, 男, (出生年 1995), 学士, 主要研究方向: 人工智能。

大型数据集的场景下, 计算效率成为这一模型首要考虑的问题。

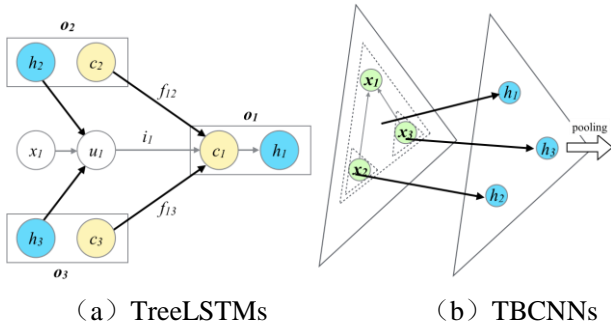


图 1 TreeLSTMs 和 TBCNNs 模型示例

相对于 TreeLSTMs, TBCNNs^[6,7]是一种针对句法树的 CNN 模型, 它的树卷积方法能实现在句法树上的并行化特征提取, 训练效率对比 TreeLSTMs 有很大提升。但由于池化操作的空间不变性假设, 模型无法在节点间对特征进行组合(故在图 1(b)中, h_1 、 h_2 和 h_3 间无任何箭头连接), 导致模型未能充分利用输入序列的结构信息。

由于 TreeLSTMs 和 TBCNNs 两种模型存在着互补特性, 本文提出一种针对句法树的混合神经网络模型。该模型以 TreeLSTMs 为改进对象, 借助 TBCNNs 的树卷积和池化方法实现了类似 TreeLSTMs 的计算, 故将此模型命名为 Quasi-TreeLSTMs。模型包含卷积模块和池化模块两个子模块, 前者完成非线性变换层和门状态的计算, 后者完成剩余的空间关联性的计算。由于池化模块的计算不存在任何参数, 因此该模块的计算耗时可忽略不计。两模块一个为模型带来了并行性, 另一个则保证了它仍然拥有和 TreeLSTMs 一样的记忆和组合特征的能力。本文将在情感分类和语义相似性两种自然语言处理任务上对模型进行测试, 实验结果表明 Quasi-TreeLSTMs 的表现普遍优于 TreeLSTMs。

本文其他部分组织如下: 第 2 节概述 TreeLSTMs 模型; 第 3 节介绍本文提出的两种 Quasi-TreeLSTMs 模型, *Dependency* Quasi-TreeLSTMs 和 *Constituency* Quasi-TreeLSTMs; 第 4 节对实验结果进行讨论和分析; 第 5 节介绍相关工作; 第 6 节总结全文。

2 背景知识

考虑到本文的改进对象是 TreeLSTMs, 本节将对此模型进行概述。图 2 展示了将 *Fruit flies like a banana* 进行解析后的两种句法树。

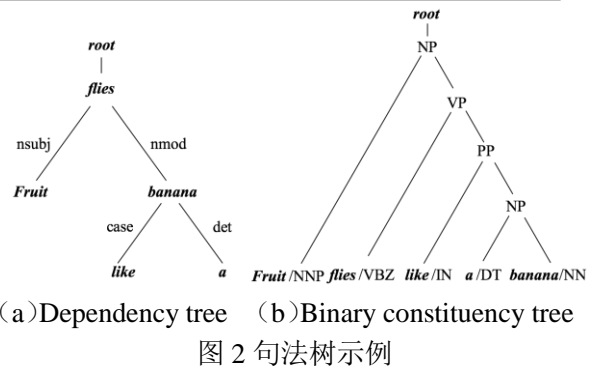


图 2 句法树示例

2.1 针对依存树建模的 *Dependency* TreeLSTMs

针对依存树建模的 TreeLSTMs 模型, 称为 *Dependency* TreeLSTMs。依存树是按照词与词间的句法关系将各个词节点相互连接的句法树, 如图 2(a)中 *flies* 和 *Fruit* 由 *nsubj* (主谓关系) 标签连接, *flies* 和 *banana* 则由 *nmod* (复合名词修饰关系) 标签连接。考虑到依存树中每个节点包含的子节点的数量各不相同(有时甚至差异巨大); 同时, 各个子节点间也不存在任何顺序, 因此, *Dependency* TreeLSTMs 在组合各子节点的隐藏状态时采用的方式是全部求和。对某个节点 j , 该模型通过如下公式计算它的隐藏状态 h_j :

$$\tilde{h}_j = \sum_{k \in C(j)} h_k \quad (1)$$

$$i_j = \sigma(W^{(i)} x_j + U^{(i)} \tilde{h}_j + b^{(i)}) \quad (2)$$

$$f_{jk} = \sigma(W^{(f)} x_j + U^{(f)} h_k + b^{(f)}) \quad (3)$$

$$o_j = \sigma(W^{(o)} x_j + U^{(o)} \tilde{h}_j + b^{(o)}) \quad (4)$$

$$u_j = \tanh(W^{(u)} x_j + U^{(u)} \tilde{h}_j + b^{(u)}) \quad (5)$$

$$c_j = i_j \odot u_j + \sum_{k \in C(j)} f_{jk} \odot c_k \quad (6)$$

$$h_j = o_j \odot \tanh(c_j) \quad (7)$$

其中 $C(j)$ 是节点 j 所有子节点的集合。

公式(2)–(5)分别代表输入门(input gate)、遗忘门(forget gate)、输出门(output gate)和非线性变换层, 其中遗忘门需要区分各个子节点 k 。它们各自有一组 (W, U, b) 共享变量, 可通过训练进行学习获得。如前文所述, 节点 j 的门状态和线性变换层的计算都依赖公式(1)组合其所有子节点的隐藏状态, 这种空间的关联性计算正是 TreeLSTMs 模型难以并行处理数据的根本所在。

2.2 针对 N 元支持树建模的 *Constituency* TreeLSTMs

针对 N 元支持树(下面统称为支持树)建模的

TreeLSTMs 模型, 称为 *Constituency TreeLSTMs*。与依存树不同, 支持树的叶子节点有序地表示输入序列中的词, 而非叶子节点代表的是短语, 连接各节点的边上也没有关系标签。如图 2 (b) 中第 2 层的非叶子节点 NP (名词性短语) 指的是 a banana, 第 3 层 PP (介词性短语) 再加入叶子节点 like, 代表 like a banana。直觉上, 支持树似乎更加符合人们由下至上组合语义的要求。考虑到支持树各个非叶子节点包含的子节点数目都不超过 N 个, 且各子节点间存在着词序。例如, 图 1 (b) 表示一个 binary constituency tree, 即二叉支持树, 图中 NP 代表的是 a banana 而非 banana a。因此 *Constituency TreeLSTMs* 在组合子节点的隐藏状态时采用的方式是线性加权。对某个节点 j , 该模型通过如下公式计算它的隐藏状态 h_j :

$$i_j = \sigma(W^{(i)}x_j + \sum_{l=1}^N U_l^{(i)}h_{j_l} + b^{(i)}) \quad (8)$$

$$f_{jk} = \sigma(W^{(f)}x_j + \sum_{l=1}^N U_{kl}^{(f)}h_{j_l} + b^{(f)}) \quad (9)$$

$$o_j = \sigma(W^{(o)}x_j + \sum_{l=1}^N U_l^{(o)}h_{j_l} + b^{(o)}) \quad (10)$$

$$u_j = \tanh(W^{(u)}x_j + \sum_{l=1}^N U_l^{(u)}h_{j_l} + b^{(u)}) \quad (11)$$

$$c_j = i_j \odot u_j + \sum_{l=1}^N f_{j_l} \odot c_{j_l} \quad (12)$$

$$h_j = o_j \odot \tanh(c_j) \quad (13)$$

以公式 (8) 为例, $U_l^{(i)}$ 代表的是子节点 l ($l = 0, 1, 2, \dots, N$, N 是节点 j 包含的子节点总数) 隐藏状态 h_{j_l} 的权值。可以看到, *Constituency TreeLSTMs* 与 *Dependency TreeLSTMs* 的计算过程基本类似, 故本节不再赘述。为了减少模型参数 (特别是 U), 本文在实验中只使用二叉支持树, 即每个非叶子节点只包含左右子节点的支持树。

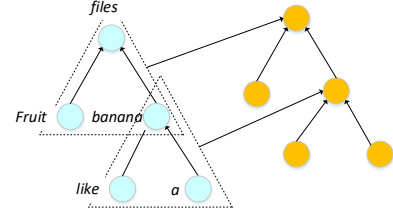
3 混合神经网络模型 Quasi-TreeLSTMs

受混合神经网络^[8,9]的启发, 本文提出一种针对句法树的 Quasi-TreeLSTMs 模型, 借助 TBCNNs 的思想, 将影响 TreeLSTMs 效率的空间关联性计算任务进行拆分, 并设计两个子模块—卷积模块和池化模块分别处理。

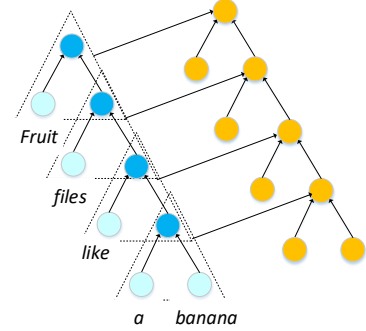
3.1 卷积模块

本文中卷积模块的任务不是直接提取特征, 而是对 TreeLSTMs 的非线性变换层和门状态进行计算。首先, 本文使用 Stanford Neural Network Dependency Parser^[10]和 Stanford PCFG Parser^[11]分

别将序列解析成依存树或支持树, 两种树结构对应 Quasi-TreeLSTMs 的两个变体 *Dependency Quasi-TreeLSTMs* 和 *Constituency Quasi-TreeLSTMs*, 分别如图 3 (a) 和 (b) 所示。



(a) *Dependency Quasi-TreeLSTMs*



(b) *Constituency Quasi-TreeLSTMs*

图 3 Quasi-TreeLSTMs 的两个变体

接着, 设计一个深度固定为 h (本文 $h=2$) 且包含 m 个卷积核 (kernels) 的卷积窗口, 让它在整棵树上滑动, 过程中对窗口内的子树进行计算。假设现在窗口内的子树上有 t 个节点, 每个节点被赋予一个向量 $x_k \in \mathbb{R}^n$ 。如果是依存树, 向量指的是节点词的词向量; 如果是支持树, 考虑到非叶子节点上没有对应的词, 在实验中将为每个非叶子节点初始化一个 n 维正态分布的随机向量。此时, 卷积窗口的输出如下:

$$I = \sigma(\sum_{k=1}^t W_k^{(i)}x_k + b^{(i)}) \quad (12)$$

$$F = \sigma(\sum_{k=1}^t W_k^{(f)}x_k + b^{(f)}) \quad (13)$$

$$O = \sigma(\sum_{k=1}^t W_k^{(o)}x_k + b^{(o)}) \quad (14)$$

$$U = \tanh(\sum_{k=1}^t W_k^{(u)}x_k + b^{(u)}) \quad (15)$$

其中, $W_k^{(i)}$ 、 $W_k^{(f)}$ 、 $W_k^{(o)}$ 、 $W_k^{(u)} \in \mathbb{R}^{n \times m}$ 是各节点 x_k 对应的权重矩阵, $b^{(i)}$ 、 $b^{(f)}$ 、 $b^{(o)}$ 、 $b^{(u)} \in \mathbb{R}^m$ 是偏置向量。

依存树每个节点包含的子节点数目不固定, 因此 *Dependency Quasi-TreeLSTMs* 可以像

TreeLSTMs 一样对全部子节点的词向量求和 (公式 (1)), 也可以利用 TBCNNs 的方法, 根据父子节点间的句法关系标签 (如图 2(a) 中的 *nsubj* 和 *nmod* 等) 为子节点分配权值矩阵, 本文将选择前者。假设此时窗口内子树的根节点为 x_j , 它的子节点数为 $C(j)$, 具体卷积计算过程如下:

$$\tilde{x}_j = \sum_{k \in C(j)} x_k \quad (16)$$

$$i_j = \sigma(W^{(i)}x_j + U^{(i)}\tilde{x}_j + b^{(i)}) \quad (17)$$

$$f_{jk} = \sigma(W^{(f)}x_j + U^{(f)}x_k + b^{(f)}) \quad (18)$$

$$o_j = \sigma(W^{(o)}x_j + U^{(o)}\tilde{x}_j + b^{(o)}) \quad (19)$$

$$u_j = \tanh(W^{(u)}x_j + U^{(u)}\tilde{x}_j + b^{(u)}) \quad (20)$$

上述计算与公式 (1) — (5) 十分相似, 训练参数的个数也相同, 但由于获取 x_k 不需任何前期计算, 打破了原本的空间关联性限制, 使这部分计算得以并行化处理。

类似地, 针对支持树的 *Constituency* Quasi-TreeLSTMs 的卷积计算过程如下:

$$i_j = \sigma(W^{(i)}x_j + \sum_{i=1}^N U_i^{(i)}x_{j_i} + b^{(i)}) \quad (21)$$

$$f_{jk} = \sigma(W^{(f)}x_j + \sum_{i=1}^N U_{ki}^{(f)}x_{j_i} + b^{(f)}) \quad (22)$$

$$o_j = \sigma(W^{(o)}x_j + \sum_{i=1}^N U_i^{(o)}x_{j_i} + b^{(o)}) \quad (23)$$

$$u_j = \tanh(W^{(u)}x_j + \sum_{i=1}^N U_i^{(u)}x_{j_i} + b^{(u)}) \quad (24)$$

当卷积窗口在树上完成一次遍历后, 会得到一棵特征树, 上面的每个节点保存了卷积获得的非线性变换层和门状态向量。但对于遗忘门, 即公式 (18) 和 (22), f_{jk} 是保存在它的各个子节点 k 上的。虽然卷积操作在一定程度上增加了空间复杂度, 但只要按批训练时 *batch* 大小设置合适, 一般不会对训练产生任何影响。

3.2 池化模块

在通过卷积模块获得的特征树上, 池化模块要完成空间关联性计算任务。可以看到, 公式 (6) (7) 和公式 (12) (13) 的计算虽然依赖前一层的计算结果, 但过程中不存在任何需要训练的参数, 对现在大多数的 CPUs/GPUs 来说计算任务不大。与 TBCNNs 中提到的 *Dynamic Pooling*^[12] 不同, 本文将采用一个和卷积窗口类似的池化窗口 (深度为 2)。因为深度固定可保证计算只在特征树的父子节点间完成, 而不会在层级间跳跃破坏空间关联性。计算过程同公式 (6) (7) 和 (12) (13), 故此不

再赘述。

3.3 目标函数构造

情感分类和语义相似性两种任务在目标函数的构建上有所差别。对于情感分类任务, 由于每个节点 j 都有一个真实标签 y_j , *softmax* 分类器需要在各个节点上完成一次分类计算获得预测标签 \hat{y}_j , 最后对训练参数 θ 有如下目标函数:

$$J(\theta) = -\frac{1}{m} \sum_{k=1}^m y_j \log \hat{y}_j + \frac{\lambda}{2} \|\theta\|^2 \quad (25)$$

其中 m 是训练样本中节点数目, λ 是 L2 正则化项超参。

语义相似性任务是两个句子的匹配问题, 需要构建两棵树并分别用一个 Quasi-TreeLSTMs 进行编码。实验中将两树在根节点上的特征向量对 (h_L, h_R) , $h_{L,R} \in \mathbb{R}^m$ 进行如下组合, 获得一个匹配向量 $h_s \in \mathbb{R}^{4m}$:

$$h_s = [h_L; h_R; h_L - h_R; h_L \circ h_R] \quad (26)$$

其中 “;” 是 *concatenation* 操作, “-” 是 *element-wise difference* 操作, “ \circ ” 是 *element-wise product* 操作。最后将匹配向量 h_s 喂入 *softmax* 分类器获得两个句子的预测标签 \hat{y} 。目标函数同公式 (25), 但 m 代表的是训练样本中句子对的个数。

4 实验分析

实验选择以下两个自然语言处理任务来测试本文提出的 Quasi Tree-LSTMs 模型的性能: (1) 情感分类, 分析上万条带标签的电影评论的情感倾向; (2) 语义关系, 根据语义关系判断句子对是否相似。

4.1 情感分类

数据集介绍 本实验使用 *Stanford Sentiment Treebank* 数据集^[13], 该数据集包含一万多条电影评论, 所有评论都带有一个人工标记的情感标签, 分别如下: 强积极 (++), 积极 (+), 中立 (*neutral*), 消极 (-), 强消极 (--). 本文在五分类和二分类两种设置上进行实验: 对五分类任务, 按 8544/1101/2210 的比例将数据集划分成训练集/验证集/测试集; 对二分类任务, 实验中不考虑标记为中立的样本, 将强积极和积极归为一类 (+), 消极和强消极为另一类 (-), 最后训练集/验证集/测试集的比例是 6920/872/1821。

训练参数 电脑配置 Intel/Xeon E5-2683V3 14 核 28 线程, NVIDIA GTX1080 显卡, 32GB 内存, 并使用 *Tensorflow* 深度学习框架实现。

本文的模型初始化和 TBCNNs 的设置相同,并在验证集上对模型进行超参调优,最后得到如下的训练参数:模型使用带有 300 个卷积核且深度为 2 的卷积窗口和在 Wikipedia 2014 和 Gigaword 5 上训练获得的 300 维 ($d=300$) 的 Glove 词向量^[14]来初始化词向量,若遇到未包含的词则赋予 300 维正态分布的随机向量。采用 AdaGrad^[15] 随机梯度下降算法,初始学习率 0.05, batch size 25, 本文对 Embedding 也进行训练,初始学习率 0.02。L2 正则化系数为 0.0001, 同时将 Embedding 层和 output 层使用的 dropout(keep probability)分别设置为 0.6 和 0.8。为了不让训练样本间波动过大,本文预先按句子长度对数据集进行排序。

实验结果 由于无法得到一些重要训练参数,尽管经过细致的调优过程,本文对 TreeLSTMs 和 TBCNNs 两个模型的实现在准确度上仍低于文献[5]和文献[6]中给出的结果。因此,参照本文对 TreeLSTMs 模型和 TBCNNs 模型的实验获得的结果来评估模型性能,与本文提出的 Quasi-TreeLSTMs 模型进行比较。表 1 展示了模型在五分类和二分类两个情感分类任务上分别训练 10 个 epochs 后的结果。

表 1 Stanford Sentiment Treebank 测试结果

编号	模型	五分类测试集准确度 (%)	二分类测试集准确度 (%)
1	<i>Dependency</i> TreeLSTMs	48.4	85.7
2	-Our Implementation	48.1	85.4
3	<i>Constituency</i> TreeLSTMs	51.0	88.0
4	-Our Implementation	50.8	87.6
5	<i>Dependency</i> TBCNNs	51.4	87.9
6	-Our Implementation	50.5	87.1
7	<i>Constituency</i> TBCNNs	50.4	86.8
8	-Our Implementation	50.0	86.6
9	<i>Dependency</i> Quasi-TreeLSTMs	50.3	87.0
10	<i>Constituency</i> Quasi-TreeLSTMs	50.5	87.3

由表 1 可知, *Dependency* TreeLSTMs 的准确度明显较低, 如果将实验结果归结到依存树上, 可以看出 *Dependency* TBCNNs 的准确度并不受影响。原因是支持树更能满足 TreeLSTMs 层级间组合信息的需求, 在处理数据的过程中细粒度在不断增大, 这对特征信息的提取非常有利, 但在依存树上缺少这一特性。TreeLSTMs 模型基于依存树训练得到的准确度不高是因为依存树结构中能训练的带标签的节点要比支持树中少了将近一半 (150k:319k), 因此能获得的信息就更少。依存树结构要比支持树结构更加紧凑, 这对 TBCNNs 的卷积和池化操作都更加有利, 因此它能在已有标签的节点上提取更多的信息。

在情感分类的两个任务上, *Dependency* TBCNNs 的准确度都比 *Constituency* TBCNNs 高了近 1%, TBCNNs 在依存树上处理叶子节点融合问题时, 根据引入的 15 个高频句法标签为子节点分配权值, 由于 TreeLSTMs 中并未这样处理, 因此本文实现 *Dependency* TBCNNs 模型时并未加入句法标签权重。虽然不考虑句法标签信息的 *Dependency* TBCNNs 的准确度在两个任务上分别下降了 0.9% 和 0.8%, 但仍高于 *Constituency* TBCNNs 的准确度, 这一实验结果也表明了 TBCNNs 在处理依存树结构的数据上有一定的优势。

虽然本文提出的 *Constituency* Quasi-TreeLSTMs 在两个任务上的准确度都高于 *Dependency* Quasi-TreeLSTMs, 这和 TreeLSTMs 的结果一样, 不同的是本文的两个模型差距却不大, 说明 Quasi-TreeLSTMs 作为一种混合模型, 缓解了已有的模型对树结构存在的敏感性, 特别是有效避免了 TreeLSTMs 在依存树上无法有效提取信息的问题。

虽然实验中的三类模型没有绝对的最优, 除 *Dependency* TreeLSTMs 外, 其他模型在五分类任务上的准确度最高和最低间相差 0.8%, 在二分类上相差 1.0%, 表现最好的是 *Constituency* TreeLSTMs, 其次是本文提出的 *Constituency* Quasi-TreeLSTMs, 这两个模型在两个任务上相差了 0.3%, 主要原因是本文提出的模型所使用的池化模块不能像 TreeLSTMs 模型那样很好地适应支持树; *Dependency* Quasi-TreeLSTMs 的准确度高于 *Constituency* TBCNNs, 说明同样是在不适合的树结构上使用模型, 因为 Quasi-TreeLSTMs 的混合特性, 能弱化这种结构和需求的不对称性。

图 4 给出了模型在情感分类的两个任务训练上每个 epoch 的平均消耗时间。TreeLSTMs 和 TBCNNs 模型实现的训练速度差异很大, 且与针对的树结构相关。表 2 列举了 TreeLSTMs 针对支持树结构在三种实现方法中平均每秒解析树的个数。

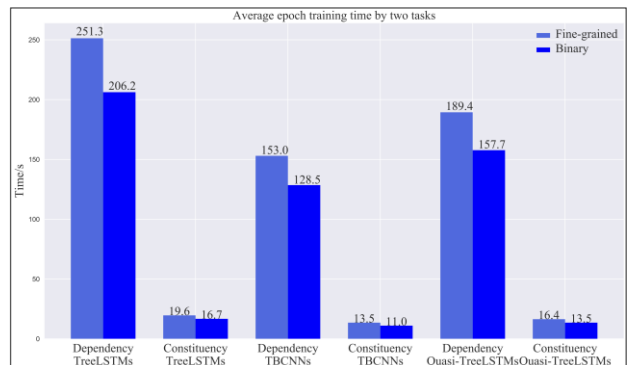


图 4 在两个任务上训练 epoch 的平均消耗时间

表2 TreeLSTMs 基于支持树的三种实现平均每
秒解析树的个数

动态图	静态图	mini-batch	mini-batch + meta-tree
1.5	9	33	420

由于 Tensorflow 本身不支持动态图模型的构建, 导致模型无法进行 batch 训练, 因此动态图和静态图的解析比较慢。在三种实现中, 动态图使用后序遍历操作, 静态图将树以列表形式存储, 并使用 Tensorflow 的 while loop 操作。mini-batch 方法^[16]基于广度优先遍历搜索变量, 类似于树的层次遍历, 而 mini-batch with meta-tree 方法^[17]在 mini-batch 基础上将 batch 中各棵树上的节点按照结构排序并融合到一棵 meta-tree 上, 然后一层一层的处理。Mini-batch meta-tree 方法的时间复杂度由 $O(M \times N)$ 缩减到 $O(\log(N))$, 训练速度非常快(这种方法同 Google 提出的 dynamic batching 算法^[18]类似)。本文基于支持树的模型都是按照 mini-batch meta-tree 的方式实现。对于依存树, 由于每个节点的子节点数目各不相同, 很难构建一棵 meta-tree, 因此只能以 mini-batch 的方式实现该模型, 因此在图 4 中, 针对依存树建模的模型在训练速度上明显低于针对支持树生成的模型。

如图 4 所示, 本文提出的 Quasi-TreeLSTMs 是混合模型, 它的训练耗时在 TBCNNs 和 TreeLSTMs 之间。但由于实现方式的相对高效, 使得针对支持树的模型的速度差异小于针对依存树的模型的差异。在两个任务上, Constituency Quasi-TreeLSTMs 的训练速度仅比 Constituency TreeLSTMs 快 3.2s, 这一现象说明除了优化模型本身来提升性能, 使用一个高效的方法实现模型也能在一定程度上削弱模型间特性的差异。

由于实现方式相对低效, 针对依存树的模型间的特性差异将在训练过程中不断体现。TBCNNs 模型最适应依存树结构, 且没有空间关联性关系的计算, 使得它的训练速度相对较快。而本文提出的 Quasi TreeLSTMs 模型在 TBCNNs 基础上融入了部分空间关联性计算, 但这部分任务是无训练参数的, 因此训练耗时在两个任务上平均只多 30s 左右。但和 TreeLSTMs 相比, Quasi TreeLSTMs 模型在两个任务上分别快了 61.9s 和 48.3s, 说明本文的模型在训练时有较大的速度提升。虽然 Quasi-TreeLSTMs 模型的准确度不及 TreeLSTMs, 但在多数情境下, 快速训练迭代并获得较好的结果才是任务的需求。

4.2 语义相关性

数据集介绍

本实验使用包含 9927 个句子对的 SICK (Sentences Involving Compositional Knowledge) 数据集^[19], 每个句子对被人工标记了 ENTAILMENT (蕴含)、NEUTRAL (无明显关系) 和 CONTRADICTION (矛盾) 三类中的一类。本实验按 4500/500/4927 将数据集划分为训练集/验证集/测试集。

训练参数

同情感分类实验。
实验结果 表 3 展示了所有对比模型在 SICK 数据集上训练 30 个 epochs 的准确度, 及训练一个 epoch 的平均消耗时间。

在 4.1 节中已经介绍, 由于支持树的特性, TreeLSTMs 模型处理支持树的数据更有优势。而依存树按照词与词间的句法关系将各个节点连接组合而成, 并带有词与词之间的语法关系, 使得 TBCNNs 模型的卷积层和池化操作在处理依存树结构的数据上更有利。由表 3 可知 Constituency TreeLSTMs (87.5%) 模型和 Dependency TBCNNs (87.0%) 模型的准确度都较高。本文提出的 Constituency Quasi-TreeLSTMs (87.2%) 模型准确度高于 Dependency Quasi-TreeLSTMs (86.7%), 由于 Quasi-TreeLSTMs 模型是一个混合模型, 因此两者差距不大。同情感分类任务, 本文的模型缓解了已有模型对树结构类型的敏感性, 避免了 TreeLSTMs 模型在依存树上无法有效提取信息的问题, 也解决了 TBCNNs 模型无法利用支持树的层级关系的缺陷。

另一方面, 在训练上每个 epoch 的平均消耗时间 TBCNNs 的训练速度相对较快, Quasi TreeLSTMs 模型在 TBCNNs 模型的基础上引入了循环模块计算序列中词序关系, 但在该模型的循环模块中无训练参数, 因此该模型的速度和 TBCNNs 模型的速度相差不大。Quasi TreeLSTMs 模型在速度上相对于 TreeLSTMs 模型有很大提升。因此, 综合模型在两种树结构上效果的平衡关系和训练速度两个衡量指标, 本文提出的 Quasi TreeLSTMs 模型最值得考虑。

表 3 SICK 数据集测试结果

模型	测试集准确度 (%)	训练一个 epoch 的平均消耗时间 (s)
Dependency TreeLSTMs	84.7	510.7
Constituency TreeLSTMs	87.5	46.7
Dependency TBCNNs	87.0	314.1
Constituency TBCNNs	85.2	35.1
Dependency Quasi-TreeLSTMs	86.7	376.1
Constituency Quasi-TreeLSTMs	87.2	40.9

5 相关工作

目前, 针对树结构建模的模型大致分为两类: 基于循环神经网络 (RNN) 在树结构上建模^[5,20]和基于卷积神经网络 (CNN) 在树结构上建模^[6,21]。

基于循环神经网络在树上构建的模型, 除了 Tree-LSTM 模型外, Dependency RNN^[18]模型在依存树上建模, 并通过结合句子的句法依赖性来提高循环神经网络模型的性能。Dependency RNN 模型在解析树的所有路径 (即从当前节点到根节点的展开) 上都相互独立地获取所需要的依赖结构。同时保存每个节点出现在路径中的频率, 并将其倒数和学习率结合, 以防某一节点出现在多个路径中造成过度训练。此外, 还在模型中加入语法标签, 提出 Labelled Dependency RNN 模型^[18]。但该模型除了本文重点强调的效率问题, RNN 模型在处理长序列上记忆状态衰减这一特点也将是 Dependency RNN 模型要面临的一项困难。

和 TBCNNs 模型相同的 DBCNNs^[19], 该模型基于 CNN 在依存树上建模, 提取每个词特征时, 将从该词到根节点的路径上的所有节点加入计算。该做法可以提取序列中长距离信息。虽然都是通过 CNN 模型基于树结构建模, 但 TBCNNs 和 DBCNNs 卷积窗口的工作方式不同, 且两个模型都没考虑输入序列的原始顺序信息。

目前存在很多将 CNN 模型结合 RNN 模型生成的混合模型。运用 CNN 卷积操作接收长度固定的短语进行学习的混合神经网络模型^[22]将生成的特征表示用于 LSTM 模型, 进一步学习输入文本的依赖关系。文献[23]将卷积层与双向 LSTM 结合生成一个新的模型, 通过对输入信息使用卷积层来处理文本, 并通过池化函数, 以减小序列的长度, 然后将生成的特征提供给双向 LSTM 模型用于后续处理。Quasi-RNN 模型^[9]是一种将 CNN 卷积操作和 RNN 的循环操作相结合的新的自然语言处理模型, 卷积操作并行计算输入门、遗忘门和输出门信息, 在循环层递归的计算输入序列中每个时间步的细胞状态和隐藏状态, 有效的解决了 RNN 时效性问题。但上面的三种模型都是基于顺序处理输入序列, 计算得到的最终表示只包含了序列的顺序信息。

6 结论

本文提出一种基于句法树的混合神经网络模型 Quasi-TreeLSTMs, 该模型把 CNNs 的卷积层和 Tree-LSTMs 模型结合, 将影响 TreeLSTMs 效率的空间关联性计算任务进行拆分, 用 CNNs 卷积层来完成最主要也是最耗时的三个控制门状态的计算, 循环层即池化部分因为完全无参数, 恰好适合用

CNNs 池化层来完成。本文用 CNNs 实现了类似 LSTMs 的操作, 在保持后者记忆性的前提下, 又为其增添了并行性。Quasi-TreeLSTMs 最后在情感分类和语义关系两种自然语言处理任务上对模型进行了测试。通过实验结果可以看出, 基于综合因素考量, 本文提出的 Quasi-TreeLSTMs 的表现普遍优于 TreeLSTMs 和基于 CNN 模型在树结构上建模的 TBCNNs。

参考文献

- [1] Peter W. Foltz, Walter Kintsch, Thomas K Landauer. The measurement of textual coherence with latent semantic analysis[J]. Discourse Processes, 1998, 25(2-3):285-307.
- [2] Landauer T K, Dumais S T. A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge[J]. Psychological Review, 1997, 104(2):211-240.
- [3] Elman J L. Finding structure in time ☆[J]. Cognitive Science, 1990, 14(2):179-211.
- [4] Mikolov T A. Statistical Language Models Based on Neural Networks[J]. 2012.
- [5] Tai K S, Socher R, Manning C D. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks[J]. Computer Science, 2015, 5(1):: 36.
- [6] Mou L, Peng H, Li G, et al. Discriminative Neural Sentence Modeling by Tree-Based Convolution[J]. 2015.
- [7] Mou L, Yan R, Li G, et al. Backward and Forward Language Modeling for Constrained Sentence Generation[J]. Computer Science, 2016, 4(6):473-482.
- [8] Balduzzi D, Ghifary M. Strongly-Typed Recurrent Neural Networks[J]. 2016.
- [9] Bradbury J, Merity S, Xiong C, et al. Quasi-Recurrent Neural Networks[J]. 2016.
- [10] Chen D, Manning C. A Fast and Accurate Dependency Parser using Neural Networks[C]// Conference on Empirical Methods in Natural Language Processing. 2014:740-750.
- [11] Klein D, Manning C D. Accurate unlexicalized parsing[C]// Meeting on Association for Computational Linguistics. Association for Computational Linguistics, 2003:423-430.
- [12] Socher R, Huang E H, Pennington J, et al. Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection[J]. Advances in Neural Information Processing Systems, 2011, 24:801-809.
- [13] Socher R, Perelygin A, Wu J Y, et al. Recursive deep models for semantic compositionality over a sentiment treebank[C]//

- Proceedings of the conference on empirical methods in natural language processing (EMNLP). 2013, 1631: 1642.
- [14] Pennington J, Socher R, Manning C D. Glove: Global Vectors for Word Representation[C]//EMNLP. 2014, 14: 1532-1543.
- [15] Duchi J, Hazan E, Singer Y. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization[J]. Journal of Machine Learning Research, 2011, 12(7):2121-2159.
- [16] Dekel O, Ran G B, Shamir O, et al. Optimal distributed online prediction using mini-batches[J]. Journal of Machine Learning Research, 2012, 13(1):165-202.
- [17] Stulp F, Sigaud O. Many regression algorithms, one unified model: A review[J]. Neural Networks, 2015, 69:60-79.
- [18] Looks M, Herreshoff M, Hutchins D L, et al. Deep Learning with Dynamic Computation Graphs[J]. 2017. Moshe Looks, 2017
- [19] Marelli M, Bentivogli L, Baroni M, et al. Semeval2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment[J]. SemEval-2014, 2014.
- [20] Mirowski P, Vlachos A. Dependency Recurrent Neural Language Models for Sentence Completion[J]. Computer Science, 2015, 17(15):p ágs. 30-35.
- [21] Ma M, Huang L, Xiang B, et al. Dependency-based Convolutional Neural Networks for Sentence Embedding[J]. 2015:174-179.
- [22] Zhou C, Sun C, Liu Z, et al. A C-LSTM Neural Network for Text Classification[J]. Computer Science, 2015, 1(4):39-44.
- [23] Xiao Y, Cho K. Efficient Character-level Document Classification by Combining Convolution and Recurrent Layers[J]. 2016.

作者联系方式: 霍欢 上海市杨浦区军工路 516 号 200093 15601787212 huo_huan@yahoo.com

作者联系方式: 张薇 上海市杨浦区军工路 516 号 200093 13644224679 zhangwei_qianrushi@126.com