《Table-to-text Generation by Structure-aware Seq2seq Learning》论文复现

张书宁
2019011311
计95
zhang-sn19@mails.tsinghua.edu.cn

张震
2019012131
基科91
zhen-zha19@mails.tsinghua.edu.cn

1 简介

本次人工神经网络大作业我们选择了《Table-to-text Generation by Structure-aware Seq2Seq Learning》论文复现任务[1]。我们组的主要贡献如下：

- 我们阅读、充分理解、并基于jittor框架完整重现了《Table-to-text Generation by Structure-aware Seq2Seq Learning》论文和对应结果。
- 我们复现了论文中主要的BLEU-4和ROUGE-4指标，并选取样例进行实际生成和论文中的样例生成效果比对。复现过程中各项指标比论文中所给略低（约0.4百分点）
- 原仓库为tensorflow-1.0.0版本和python-2.7版本代码，相对比原仓库的代码，我们给出了新版tensorflow实现的代码，并可以在tensorflow-1.13.0上正常工作。除此以外，对于jittor和tensorflow而言，两者对于神经网络，计算图的定义方式不完全相同，在复现过程中我们着重关注了这一点，除了用jittor实现了tensorflow中更复杂的函数以外，还部分修改了tensorflow的while-loop结构，使jittor复现版本效率得到优化。对于我们复现的版本，整体数据预处理，训练部分和预测部分解耦较强，整体模型由SeqUnit包装，参数都可以传入SeqUnit，对于迁移学习和模型修改较为容易。开源链接见：https://github.com/namezhenzhang/wiki2bio

2 复现论文介绍

从结构化的表格到序列文本生成是自然语言生成（NLG）领域的重要任务。很多过往的作者基于不同的精巧结构设计了不同的针对复杂图和表格结构的隐层表示方法。《Table-to-text Generation by Structure-aware Seq2Seq Learning》文章作者基于Seq2seq模型增加了对于结构化输入的适应，更好的完成了wiki表格（结构化数据）到序列文本（非结构化数据）的生成任务。作者的主要贡献分为三部分：

设计两种不同尺度的注意力机制 首先，作者提出了基于Seq2Seq结构改进的模型，更好的融合了两种不同尺度的注意力机制，让模型能够更好抽取图表结构化信息。

在LSTM结构中增加Field-gating门 作者对LSTM和注意力机制部分进行了修改，让这两种注意力机制可以良好并高效的融合到模型中。

基于WIKI2BIO进行测试和比较 作者基于WIKI2BIO常用数据集与baseline模型进行了比较，并通过消融实验证明了所提出方法的有效性。

作者将机器对于结构化文本的转换和人对于结构化文本的转换进行了建模和关联。他认为结构化的提取和表示主要分为两个部分：local addressing和global addressing。这是因为：结构化的表格本质上是多个field（域）和value（值）的映射。对于这种机构化数据中信息的抽取就主要分为两个层次："宏观层次"是对于域之间关联，域的重要程序的提取，也就是field-level的嵌入表示；"微观部分" 是对于域内部上下文关联，内容的提取，也就是context-level，或word-level的嵌入表示。其中local addressing主要指如何建模局部的文

本之间的关联，选择表中重要的文本用于生成，global addressing指如何建模文本之间的关联，将文本更好的串联成自然语言的序列。

作者在网络中很好的融合了global addressing 和local addressing机制。他主要通过在encoder中增加对于内容的编码，基于词级别的注意力机制来实现local addressing；而作者主要通过在LSTM网络中添加field-gating门，基于field-gate级别的注意力机制来实现global addressing（见图 1）。

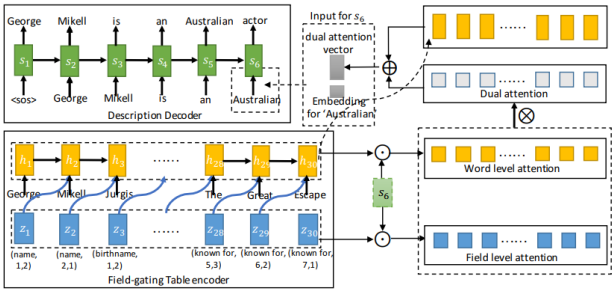

Figure 1: 模型总览

域级别的嵌入 域和值的"键值对"信息在结构化表格中不仅意味着内容信息，还隐含了相对位置信息等。作者基于Lebret等的设置[2]进一步扩展，将域中词的信息表示成三元组如下：

$$Z_\omega = \{f_\omega; p_\omega^+; p_\omega^-\}$$

其中p_ω^+表示词相对于这个域从开始计算偏移量，p_ω^-表示词相对于这个域从末尾倒算的偏移量，f_ω表示域的名字。这样即使在同一个域内有两个词名字相同，因为他们的相对位置不同，所以对应的三元组（实际上是$(field, position)$组）也会不同。作者修改了原有的LSTM结构并添加了对于域的结构化的表示方式——域门(field-gating)如下：

$$\begin{pmatrix} l_t \\ \hat{z}_t \end{pmatrix} = \begin{pmatrix} sigmoid \\ tanh \end{pmatrix} W_{2n,2n}^d (z_t)$$

$$c_t^{`} = f_t \otimes c_{t-1} + i_t \otimes \hat{c}_t + l_t \otimes \hat{z}_t$$

其中z_t 代表域嵌入，$l_t \in [0,1]^n$决定了有多少域信息应该保留，$\hat{z}_t$是域门对应的域信息，$c_t^{'}$和c_t是对应的细胞状态，其中$c_t^{'}$是由c_t融合了表的域结构信息而来（具体形式见图 2）。

name	George Mikell
birthname	Jurgis Mikelaitis
birthdate	4 April 1929 (age 88)
birthplace	Bildeniai, Lithuania
nationality	Lithuanian, Australian
occupation	Actor, writer
years active	1957–present
known for	Thc Guns of Navaronc The Great Escape

word	*Field embedding*
George	(name, 1, 2)
Mikell	(name, 2, 1)
Jurgis	(birthname, 1, 2)
Mikelaitis	(birthname, 2, 1)
4	(birthdate, 1, 7)
April	(birthdate, 2, 6)
...	...
The	(known for, 5, 3)
Great	(known for, 6, 2)
Escape	(known for, 7, 1)

Figure 2: 域信息编码

局部信息和全局信息 为了同时考虑到局部信息(local addressing)和全局信息(global addressing)，如前所述，作者使用了含有dual attention的LSTM结构。具体而言，如果用$H = h_{t\,t=1}^L$表示编码器的隐层状态，用$Z = z_{t\,t=1}^L$表示域的嵌入，那么在t时刻生成的token可以建模成关于隐层状态，域嵌入和t时刻前生成token的函数如下：

$$P(w_t|H, Z, w_{<t}) = softmax(W_s \otimes g_t) g_t = tanh(W_t[s_t, a_t]) s_t = LSTM(w_{t-1}, s_{t-1})$$

其中s_t是解码器的第t个隐层状态。a_t是注意力向量，根据VVanilla 注意力机制按照softmax加权的形式决定了隐层状态权重如下:

$$\alpha_{t_i} = \frac{e^{g(s_t,h_i)}}{\sum_{j=1}^{N} e^{g(s_t,h_j)}} a_t = \sum_{i=1}^{L} \alpha_{t_i} h_i$$

其中g函数为编码器和解码器隐层状态的相对分数，实际上用tanh函数和内积进行实现(下面W_s, W_t, W_p, W_q)均为模型参数:

$$g(s_t, h_i) = tanh(W_p h_i) \otimes tanh(W_q s_t)$$

而最后作者修改的对偶注意力部分由域级别的注意力机制决定。作者对于对偶注意力机制部分保持了和前述相同的结构和函数定义方式如下:

$$\beta_{t_i} = \frac{e^{g}(s_t, z_i)}{\sum_{j=1}^{N} e^{g(s_t,z_j)}}$$

$$g(s_t, z_i) = tanh(W_x z_i) \otimes tanh(W_y s_t)$$

$$\gamma_{t_i} = \frac{\alpha_{t_i}\beta_{t_i}}{\sum_{j=1}^{N} \alpha_{t_j}\beta_{t_j}}$$

$$a'^{t=\sum_{i=1}^{L} \gamma_{t_i} h_i}$$

实验 作者利用WIKI2BIO数据集尝试了两种不同的实验，给出了统计和测试结果，模型超参和训练细节，并提供了具体的案例分析(case study)。首先，作者和Kneser-Ney模型，基于模板生成的Kneser-Ney模型，NLM，table NLM，Vanilla Seq2seq模型对比，展示了作者所提出的模型在BLEU-4和ROUGE-4分数上有较大提升，并通过提供注意力的热力图证明作者提出的模型确实关注到了域级别，域内部词汇级别两层次的相关关系。此外，作者进一步通过打乱表格内部顺序，通过类似消融实验的形式证明即使表格内部内容顺序打乱作者也能较好建模和生成语句，这证明了global addressing和local addressing机制的有效性。作者还提供了具体的生成样例以及和baseline模型的结果比较，可以看到作者提供的生成更自然。

我们复现时参考的作者代码实现原深度学习框架为tensorflow 1.0.0, python 2.7；用jittor复现时有如下核心要点和难点:

- 原论文框架为tensorflow，基于静态图实现；jittor是实时编译框架，整体更贴近动态图实现模式。在复现时，更多的我们不仅仅需要考虑复现时语法是否相符，还需要考虑语义层面这样的复现语句是否合理、是否高效。如果按照静态图方式多定义了很多冗余参数可能就会大幅降低训练效率。
- 原论文重新定义和修改了LSTM的网络结构，我们需要对作者整体的网络拓扑结构和用意有较多的了解。除此以外，作者自己定义了对于源数据的预处理格式，一方面我们需要充分理解这种数据预处理格式，另一方面如果我们进行实时生成需要能够转换和自己实现数据预处理过程。
- jittor的整体实现和理解有一定难度。特别是jittor的计算图构建模式和实现模式和pytorch，tensorflow均有一定差别，如何提高效率也是复现过程中需要思考的问题。

3 实验

3.1 实验设置

遵照原论文的实验设置，我们基于WIKBIO数据集[2]进行了实验。我们的实验设备为32G@V100和16G@T4，其中训练过程使用32G@V100，测试过程使用16G@T4。总计训练时间约为40h，测试时间约为10h。

论文数据集介绍 WIKIBIO数据集是和人物有关的数据集，涵盖了2015年9月的728321篇文章，用每篇文章的第一句话作为infobox的描述。具体的infobox形式见图 3，数据集具体统计信息见表Table 1。复现版本中使用了论文原作者处理的数据集，保持了和原作

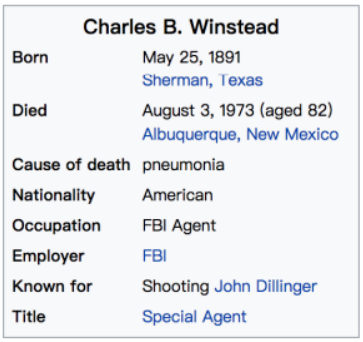

Figure 3: WIKI2BIO数据格式

	每句话的token个数	表中每句话的token个数	每张表的token个数	每张表的域个数
平均值	26.1	9.5	53.1	19.7

Table 1: WIKIBIO数据集统计信息

者相同的(8:1:1)的训练，验证，测试集比例；并和原论文保持了相同的评估指标(test指标为BLEU-4和ROUGE-4(F-measure))。

模型的超参数设置

和原论文保持一致部分 目标词表大小(target vocab size), 建模位置的词表大小(position vocab):31, 表格域词表大小(source vocab):1480, 源词表大小(source vocab size):20003, 隐层大小(hidden size):500, 嵌入大小(emb size):400, 域隐层嵌入大小(field size):50, 位置隐层嵌入大小(pos size):5 整体均和源论文保持一致，其他修改部分参数见下。

修改部分 为加快训练收敛速度，更高效利用计算资源，batch size取为64(train)和256(valid),learning rate调整为0.0003。实验发现可以较快收敛并达到较好效果。

3.2 实验结果

事实上，作者给出的指标基本可以复现。我们的结果比作者略低（约为0.3%）。我们进行了对应的原因分析。

- 我们进行了批大小(batch size)的调整。值得注意，因为原论文复现基于GeForce@1080，我们的复现版本基于V100@32G，GPU空间和资源大于原论文，在实际训练时batch size大于原论文。由此，我们尝试将超参向原论文方向修改。在减小batch size的时候可以观察到BLEU score 和ROUGE-4 score（测试指标）结果有上升，在进一步增大batch size的时候发现BLEU score和ROUGE-4 score（测试指标）结果有下降。由此，我们推断可能一部分的结果的偏差来源于batch size的大小，这可能主要是因为batch内数据分布，梯度更新方式不同造成的。但我们发现，即使调整到原论文的batch size大小，实际分数也要比原论文略低。

- 由于训练时间（40h）较长所限，我们的试验次数有限。相比于原论文结果，可能未达到最优。

除此以外，我们分析了具体的案例。我们随机从样本中选取了一个生成样例（图 4）用于展示，生成的语句与维基百科上2016年版描述完全一样。（见表 4）。[1]

[1]参考https://en.wikipedia.org/wiki/Jude_Stirling

	原论文	复现版本
Word dimension	400	400
Field dimension	50	50
Position dimension	5	5
Hidden Size	500	500
Target Vocab Size	20003	20003
Position Vocab Size	31	31
Source Vocab Size	1480	1480
Field Vocab Size	20003	20003
Batch Size	32	**64**
Learning Rate	0.0005	**0.0003**
Optimizer	Adam	Adam

Table 2: 论文的超参数设置

	BLEU-4	ROUGE-4
Field-gating Seq2Seq tf	43.74 ± 0.23	40.53 ± 0.31
+ dual attention	44.89 ± 0.33	41.21 ± 0.25
+ beam search(k=5)	44.71	41.65
Field-gating Seq2Seq jt(ours)	43.55	39.86
+ dual attention	44.30	40.26
+ beam search(k=5)	44.41	40.95

Table 3: 实验结果。

3.3 额外工作

3.3.1 beam search

原代码中的beamsearch实现不完全，且没有引入注意力权重来替换unktoken，我们将功能完整实现并达到了和论文中相近的结果

3.3.2 性能比较

在复现时，我们着重关注了tensorflow和jittor的异同。因pytorch和jittor整体相似，而tensorflow基于静态图结构，直接复现而不加转换会造成很多额外空间开销。经过对于内存空间的优化和变量初始化方面的优化，单epoch从14h降低到了3.5h，batch size从只能支持2到可以支持64(32G@V100)。进一步分析各阶段所消耗时间如下所示，经过对部分tensorflow循环的修改、循环内变量定义优化等核心模块效率约提高34%，相对比原仓库的tensorflow版本性能提升361.9%(2.3529s/batch到0.6502s/batch，batch size=32)。

Figure 4: Wiki info box样例

	生成语句
Jude_Stirling	jude barrington stirling (born 29 june 1982) is an english footballer who plays as a defender for brimsdown f.c. .

Table 4: 实验结果

	tensorflow(GPU)	tensorflow(CPU)	jittor
version	1.13.1	1.0.0	1.3.1.27
speed(s/batch)	1.4963	2.3529	0.6502

原论文基于tensorflow 1.0.0版本和python 2.7版本进行论文的实现。由于代码框架老旧，tensorflow 1.0和python 2.7均不再支持，在jittor上进行重新复现有如下重要意义:

- 首先，基于jittor复现tensorflow的文章相对较少，这种复现可以有效建立tensorflow，特别是tensorflow较低版本和jittor之间的关联和复现对应关系，有效提升jittor开源生态。此外，structure-aware的seq2seq模型不仅仅是结构化数据到自然语言变换的常用模型，也是多数自然语言生成任务的常用结构。复现这样的典型模型更便于后来者基于我们的模型加入预训练模型部分或者做更多的改进，这也有效的提升了jittor的编程速度，降低了jittor的编程门槛。
- 其次，我们给出了良好的可运行版本，后来的学习者可以根据我们的论文复现进行对应论文的学习，而不是花费更大精力研究过时的实现版本。值得注意的是，jittor的复现版本相对更接近pytorch，而pytorch的开发、扩展和对模型拓扑结构的拓展相对比tensorflow而言更轻便，效率较高，模型理解更容易，因此学习成本和运行成本较低。而且对于pytorch的静态图结构扩展更容易，学习者更容易在此论文基础上进行调参和修改。
- 最后，我们给出了有效的论文指标评价和基于jittor的性能评估、优化。这部分有利于其他开源社区内的jittor使用者共同交流如何更高效的利用jittor进行神经网络的设计、实现和训练。可以看到，我们复现的版本对于时间和内存的开销都比原论文小很多，也能很好的适应和利用GPU的资源和特性。

3.3.3 交互界面

我们利用pyqt实现了一个可交互界面。通过用户输入的infobox的label-value值对，来生成描述文本。此外，我们将模型和论文的部分介绍放在了交互界面当中，并希望未来可以在交互界面中实现和提供jittor和tensorflow的指标对比功能和实时few-shot训练参数调节过程。本交互界面在逐步完善过程中，也在github仓库中同步上传和更新稳定版本。

4 总结

本工作在jittor框架下复现了《Table-to-text Generation by Structure-aware Seq2Seq Learning》论文，在基本还原原作者实验结果的同时使效率得到较大提升，可用性和扩展性有进一步提高。未来我们会更多探索tensorflow和pytorch框架在jittor框架下的移植，适配和加速，特别是如何让tensorflow的静态图结构下效率比较高的神经网络在pytorch和jittor的动态图结构下进一步高效利用资源（这也是本次大作业细节方面探索较少的内容）。

参考文献

References

[1] Liu, T., K. Wang, L. Sha, et al. Table-to-text generation by structure-aware seq2seq learning. *CoRR*, abs/1711.09724, 2017.

[2] Lebret, R., D. Grangier, M. Auli. Generating text from structured data with application to the biography domain. *CoRR*, abs/1603.07771, 2016.