北京大学学报(自然科学版)
Acta Scientiarum Naturalium Universitatis Pekinensis
ISSN 0479-8023,CN 11-2442/N

《北京大学学报(自然科学版)》网络首发论文

题目：具有选择性局部注意力和前序信息解码器的代码生成模型
作者：梁婉莹，朱佳，吴志杰，颜志文，汤庸，黄晋，余伟浩
DOI：10.13209/j.0479-8023.2020.086
收稿日期：2020-06-08
网络首发日期：2020-10-14
引用格式：梁婉莹，朱佳，吴志杰，颜志文，汤庸，黄晋，余伟浩．具有选择性局部注意力和前序信息解码器的代码生成模型．北京大学学报(自然科学版).
https://doi.org/10.13209/j.0479-8023.2020.086

北京大学学报(自然科学版)
Acta Scientiarum Naturalium Universitatis Pekinensis
doi: 10.13209/j.0479-8023.2020.086

具有选择性局部注意力和前序信息解码器的代码生成模型

梁婉莹[1,2] 朱佳[1,2,†] 吴志杰[1,2] 颜志文[1,2] 汤庸[1,2] 黄晋[1,2] 余伟浩[1,2]

1. 华南师范大学计算机学院, 广州 510631; 2. 广州市大数据智能教育重点实验室, 广州 510631;
† 通信作者, E-mail: jzhu@m.scnu.edu.cn

摘要 代码生成是机器翻译领域的重要研究方向。目前已有的代码生成模型虽性能较好, 但包含的前序信息仍不足, 且缺乏对上下文相关向量计算范围的限定。本文提出了一种基于语法的代码生成模型, 该模型具有选择性局部注意力和包含前序信息的长短期记忆神经网络解码器, 通过更改上下文向量的计算范围和在解码过程中融合更多的前序信息来增强单词之间的相关性。在两个著名的数据集 Hearthstone 和 Django 进行的代码生成实验验证了本模型的有效性, 与最新模型相比, 我们的模型不仅表现出更出色的准确率和双语评估学习成绩, 还使计算工作量最小化。

关键词 代码生成; 抽象语法树; 包含前序信息的长短期记忆神经网络; 选择性局部注意力

Syntax-based Code Generation Model with Selective Local Attention and Pre-order Information LSTM Decoder

LIANG Wanying[1,2], ZHU Jia[1,2,†], WU Zhijie[1,2], YAN Zhiwen[1,2], TANG Yong[1,2], HUANG Jin[1,2], YU Weihao[1,2]

1. School of Computer, South China Normal University, Guangzhou 510631; 2.Guangzhou Key Laboratory of Big Data and Intelligent Education, Guangzhou 510631; † Corresponding author, E-mail: jzhu@m.scnu.edu.cn

Abstract Code generation is an important research direction in the field of machine translation. Although the existing code generation models have good performance, it contains insufficient pre-order information and is lack of consideration for the calculation of contextual relevance. In this article, we propose a syntax-based code generation model with selective local attention and a pre-order information decoder based on long-short term memory neural network, which aims to enhance the relevance by changing the calculation scope of the context vector and fuse more pre-order information during the decoding process. Code generation experiments in two well-known dataset, Hearthstone and Django, verify the validity of the model. Compared with state-of-the-art models, our model not only achieves excellent accuracy and bilingual evaluation understudy score, but minimizing computational effort.

Key words code generation; abstract syntactic tree; pre-order information LSTM; selective local attention

国家自然科学基金(61877020,U1811263,61772211)、广东省重点领域研发计划(2018B010109002)、广州市科学技术计划项目(201904010393)和广州市大数据智能教育重点实验室(201905010009)资助

收稿日期: 2020–06–08; 修回日期: 2020–08–08

代码生成是机器翻译的一个分支问题，它比自然语言生成或翻译[1]要求更高。只有当代码符合规定的文法以及格式,才能被机器真正识别并运行。随着机器学习领域的发展，人们逐渐找到了代码生成的解决方案。最初，Brown 提出了仅依赖于统计分析的统计机器翻译模型(Statistical Machine Translation,SMT)[2]，虽然它的准确率相对较低，但它是当时代码生成问题的新突破。随后，Bahdanau 提出了基于 Seq2Seq[3]框架的神经机器翻译模型(Neural Machine Translation,NMT)[4]。该模型可以从自然语言序列映射到目标代码序列，支持可变长度的输入和输出，并在翻译、对话和单词概括中表现出色。之后，一些研究人员尝试使用指针网络来改进 NMT 模型，例如致力于解决输出严重依赖于输入的问题的潜伏预测器网络(Latent Predictor Networks,LPN)[1]、通过长短期记忆单元(Long-Short Term Memory,LSTM)[5]生成多层树的代码逻辑形式的 Seq2Tree 模型[6]、基于句法神经网络的 Seq2Seq 模型(Syntactic Neural Network Model, SNM)[7]、抽象语法网络(Abstract Syntax Networks, ASN)模型[8]、基于语义分析和代码生成的抽象解析器(TRANX)[9]，采用 Transformer[10]架构并捕获长依赖性的句子 TreeGen[11]模型以及基于语法结构的卷积神经网络(CNN decoder model)解码器模型[12]。这些年来，前辈们在代码生成问题上取得了巨大的成功，上述模型在双语评估研究(BLEU)和准确率上逐渐得到了提高。

但上述模型在代码生成方面仍存在以下缺陷。首先，传统的序列到序列(Seq2Seq)模型(例如 LPN 模型[1]和 SNM 模型[7])的解码器所包含的结构信息较少，可能会导致前序信息对当前运算的影响不足，使得当前向量在抽象语法树中位置选择错误。其次，单纯使用软注意力，即将所有编码器输出向量置入上下文向量的计算中，会导致输入序列的先验语义向量被后验语义向量所覆盖。它缺乏了对输出语法和模型特征[1]之间模块化或紧密耦合的考虑。

针对以上问题，本文提出了一种基于语法的代码生成模型，该模型具有选择性局部注意力和带有前序信息的长短期记忆单元解码器，即 PI-LSTM(Pre-Order Information LSTM)。本文模型使用序列到序列(Seq2Seq)的编码–解码框架。在编码器部分，我们使用门控循环单元(Gate Recurrent Unit, GRU)[13]作为计算单元。与 SNM 模型中的朴素 LSTM 相比，GRU 相对简单高效，在减少训练时间的同时对准确率或双语评估学习成绩 BLEU 分数的影响非常小，适用于训练大规模预处理数据。在注意力机制部分，为了增强上下文之间的相关性，我们使用选择性局部注意力。选择性局部注意力通过限制滑动窗口中上下文向量的计算范围，减少了无关单词向量对预测单词向量的干扰。在解码器部分，为了包含更多的结构信息，我们使用基于朴素 LSTM 创新的 PI-LSTM 单元作为解码器计算单元。PI-LSTM 涵盖更多的前序信息并对输出进行二次更新，减少了模型在抽象语法树中生成错误分支的可能。

本文模型的贡献点如下。

1) 我们提出并使用选择性局部注意力机制。选择性局部注意力可以根据预设或输入语句的长度采用不同的注意力机制，从而可以减少无关单词对当前单词权重计算的干扰。相比单纯使用软注意力，选择性局部注意力可以更好地适应不同大小的数据集。

2) 我们提出并使用基于朴素 LSTM 改进的 PI-LSTM 作为解码器计算单元。通过对输出值的二次更新，它提升了前序信息在解码运算中的参与度，更好地修正了解码运算。本文模型在准确率和 BLEU 分数方面的表现均比其他模型出色。

3) 我们使用 GRU 作为编码器计算单元。与朴素 LSTM 相比，简单高效的 GRU 在对准确率等产生可忽略的负面影响的同时，大大减少了编码器计算资源的开销。

1 代码生成相关工作

1.1 SNM 模型

基于句法神经网络的 Seq2Seq 模型(SNM)[7]是目前最具代表性的代码生成模型。SNM 模型不必恢复原

有的基础语法，只需要集中精力学习现有语法规则之间的可组合性[7]。它配备了标准的递归神经网络(RNN)[14]解码器，以允许更多的神经连接反映抽象语法树的递归结构。SNM 模型应用双向 LSTM(Bidirection-LSTM)[15]编码器、LSTM 解码器和软注意力来训练给定的预处理数据集，与 2017 年之前推出的模型相比，SNM 模型的准确率非常出色。

1.2 抽象语法树

抽象语法树(Abstract Syntax Tree, AST)[16]是领域专用语言(即代码)的树型结构。AST 仅用作代码摘要，并不代表实际语法的每个细节,其每个节点都代表源代码中的一个分支。抽象语法树有两个特征，其一是它们不依赖于具体的语法。在解析输入语句时，系统会构造相同的语法树，且该语法树给编译器后端提供一个清晰、统一的接口。另一个是它们构造语法树时不依赖于语言的细节，并针对不同的语言有不同的表示形式[7]。

2 编码器–解码器代码生成模型

代码生成是不定长输入到不定长输出的转换过程。本文提出了一种基于语法的代码生成模型，如图 1 所示，该模型具有 GRU 编码器、选择性局部注意力以及 PI-LSTM 解码器。该模型根据输入序列的第一个单词生成目标抽象语法树的主干，而后通过编码器–解码器的计算逐步将语法树主干填充成完整语法树。编码器对输入序列中的每个单词进行编码，并将编码输出传送到选择性局部注意力计算上下文向量。解码器接收到上下文向量后使用 PI-LSTM 计算并解码而得到最终输出。

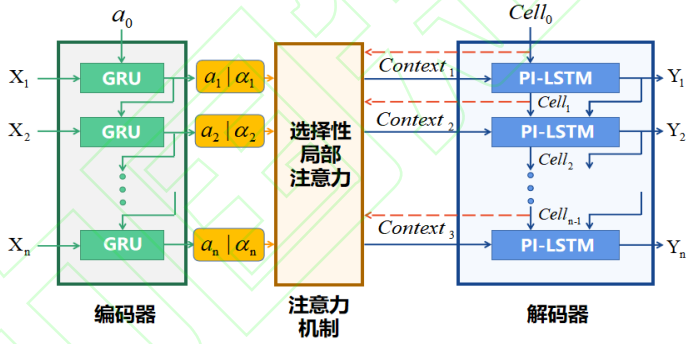

图 1　基于语法的选择性局部注意力和 PI-LSTM 解码代码生成模型

Fig. 1　A Syntax-based Code Generation Model with Selective Local Attention and PI-LSTM Decoder

2.1 底层语法驱动

代码生成模型多数是以抽象语法树(AST)为基础的。在计算机科学和语言学中，句法分析(或语法解析)[17]是一种分析和确定的过程，也是由一系列单词组成的输入文本的语法结构。底层语法驱动程序包含一组生产规则,并通过固有的几个生产规则生成头节点以及多个子节点生成相应树结构。若输入序列中的第一个单词是“if”，模型则将以“If→expr [test] stmt * [body] stmt * [or else]”的形式生成主干 AST。然后，模型将使用 GENTOKEN 和 APPLYRULE 分别在非终端节点和终端节点上进行操作，前者通过添加终端令牌来填充可变终端节点，后者将生产规则应用于当前派生树，最终达成扩展和填充空值语法树的目的。

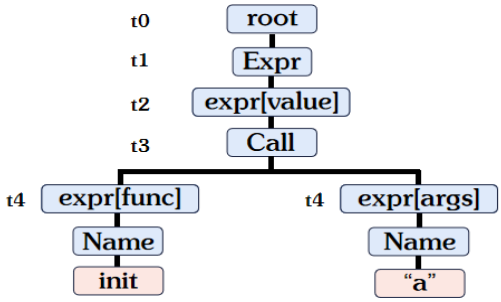

图 2　代码 init(a)的抽象语法树构造

Fig. 2　The Abstract Syntax Tree Structure of code: init(a)

2.2 门控循环单元(GRU)编码器

为了减少模型的训练时间，在编码器中，我们以门控循环网络 GRU 作为计算单元。GRU 的门控通过丢弃或保留输入序列数据，来调整信息流并沿序列传递相关信息，以预测下一个输出。GRU 是 LSTM[5]的一种变体，它将遗忘门和输入门简化为单个更新门，使计算相对简单，更适合于训练大规模预处理数据。给定输入序列中的每个单词向量$X_i = \{X_1, X_2, \ldots, X_n\}$，我们通过 GRU 计算转换为编码向量$a_i$ (默认a_0是零向量)。具体步骤如下:

首先，单词向量X_i和前序编码输出向量a_{i-1}通过 Sigmoid 函数[19] 计算获得更新门的结果z_i和重置门的结果r_i，计算公式如(1)(2)所示。

$$z_i = Sigmoid(W_z \cdot [a_{i-1}, X_i]) \tag{1}$$

$$r_i = Sigmoid(W_r \cdot [a_{i-1}, X_i]) \tag{2}$$

然后，单词向量X_i、前序编码向量a_{i-1}和复位门结果r_i输入 tanh 函数计算当前记忆内容$\tilde{h}_i$，如公式(3)所示。

$$\tilde{h}_i = \tanh(W \cdot [r_i \cdot a_{i-1}, X_i]) \tag{3}$$

最后，模型将更新门结果z_i、前序编码向量a_{i-1}和当前记忆内容$\tilde{h}_i$放入公式(4)计算，以获得最终记忆内容，即当前编码输出a_i。其中，公式(1)(2)(3)中的W_z、W_r和W均为待学习的权重参数，其最优值将通过神经网络训练获得。

$$a_i = a_{i-1} \cdot (1 - z_i) \cdot + z_i \cdot \tilde{h}_i \tag{4}$$

2.3 选择性局部注意力

在代码生成问题中，注意力机制[21]是相对重要的一环。在研究中我们发现，无论输入序列的长短，SNM 模型所使用的软注意力将所有编码输出匹配度α以及编码输出a加入上下文相关度计算，并通过 Softmax 函数[20]将其归一化。具体地，经过 2.2 节系列公式计算后，在每个解码时间步 j 中，模型会将编码后的向量a_i和前序解码单元状态Cell_{j-1}连接在一起(章节 2.4 将对Cell_{j-1}进行解释)。经过下面公式(5)和(6)的计算以及 softmax 函数归一化，我们将获得编码输出向量a_i之间的匹配度α_{ij}, W_a为待学习的权重参数。

$$e_{ij} = \tanh\left(W_a\left[Cell_{j-1}; a_i\right]\right) \tag{5}$$

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_X} exp\,(e_{ik})} \tag{6}$$

这种单纯使用软注意力的计算方式使用较多，但会导致在长句翻译中单词之间的相关性差值被削弱而不再明显。被偏置的相关性值将影响当前解码输出，从而导致生成目标代码的准确率下降。因此，我们参考其他注意力机制，如局部注意力机制，如图 3 所示。与软注意力相比，局部注意力使用滑窗限制上下文向量的计算范围，在长句的表现上明显更好。但单纯使用局部注意力仍将导致在短输入序列中语境理解能力较弱，准确率相对较低。

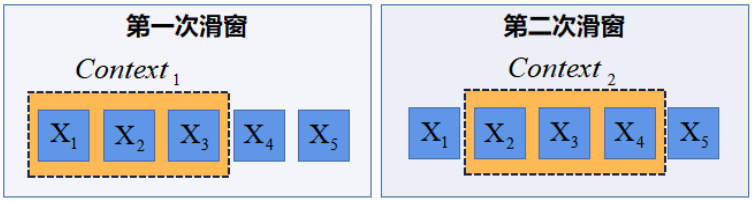

图 3　局部注意力工作示意图

Fig. 3　Schematic Diagram of Local Attention

因此，本文提出了一种用于代码生成的选择性局部注意力。模型首先判断输入序列的单词数量n，而后确定可选范围大小K。当单词数少于K时，模型将使用软注意力机制[22]，即将所有编码输出匹配度α以及编码输出a加入上下文相关度计算，如公式(7)所示。

$$Context^{<t>} = \sum_{i=1}^{n} \alpha(i,i') \cdot a^{i'}, n < K \tag{7}$$

而当单词数量n大于K时，模型将使用局部注意力的滑动窗口来限制上下文向量的计算范围，并根据当前计算位置滑动窗口。模型将对滑动窗口中所有编码输出匹配度α以及编码输出a执行加权和计算，如公式(8)所示。通过限制上下文向量的计算范围，选择性局部注意力减少了无关单词向量对当前计算单词向量的干扰。当前计算位置t在滑窗正中间，这意味着当前计算向量将与左右两边的向量参与到上下文相关度计算。如果滑窗下限小于位置 1，则从位置 1 开始。如果滑窗上限大于序列的最后位置，则以最后位置结束。滑窗的长度D等于K/2。经本文多次实验验证，当单词数大于 12 时，K 选取 12(即滑窗长度 D 为 6)可达最优效果；小于 12 则 K 选取 14(即滑窗长度 D 为 7)。

$$Context^{<t>} = \sum_{i=word_t-D}^{word_t+D} \alpha(i,i') \cdot a^{i'}, n \geq K \tag{8}$$

2.4 包含前序信息的长短期记忆网络

在解码部分，为了将更多的前序解码信息添加到网络结构中进行计算，我们使用 PI-LSTM 作为解码器计算单元，如图 4 所示。

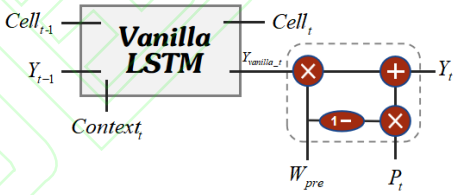

图 4　PI-LSTM 结构图

Fig. 4　Structure of PI-LSTM

PI-LSTM 是基于朴素 LSTM 改进的计算单位。模型利用 PI-LSTM 对输出进行二次更新，从而减小在抽象语法树中生成错误的分支的可能性。在每个时间步中，我们将 3 个参数，即上下文向量Context$_t$、前序 PI-LSTM 解码输出Y_{t-1}和前序解码单元状态Cell$_{t-1}$放入 PI-LSTM 获得当前解码输出Y_t和当前操作单元状态Cell$_t$。公式中的W是每个门控中实时变化的权重，最佳参数将通过模型训练获得。具体步骤如下:

首先，前序解码输出Y_{t-1}和上下文向量Context$_t$输入 Sigmoid 函数(公式(9)和(10))以计算遗忘门f_t和输入门i_t的结果。

$$f_t = Sigmoid(W_f \cdot [Y_{t-1}, Context_t] + b_f) \tag{9}$$

$$i_t = Sigmoid(W_i \cdot [Y_{t-1}, Context_t] + b_i) \tag{10}$$

然后，前序解码向量Y_{t-1}和上下文向量Context$_t$输入 tanh 函数以计算当前记忆状态$\tilde{c}_t$，如公式(11)所示。

$$\tilde{c}_t = tanh(W_C \cdot [Y_{t-1}, Context_t] + b_C) \tag{11}$$

然后，遗忘门结果f_t、输入门结果i_t和前序计算单元状态Cell$_{t-1}$将执行如公式(12)的计算以获得当前操作单元状态Cell$_t$的状态。

$$Cell_t = Cell_{t-1} \cdot f_t + i_t \cdot \widetilde{c}_t \tag{12}$$

此时, 我们需要确认最终输出。上下文向量Context$_t$进入 Sigmoid 函数(公式(13))以计算输出门结果o_t。当前操作单元状态Cell$_t$进入 tanh 函数(公式(14)), 然后与输出门结果o_t相乘, 将其作为当前 PI-LSTM 中朴素 LSTM 输出的结果$Y_{\text{vanilla_t}}$。

$$o_t = Sigmoid(W_o \cdot [Y_{t-1}, Context_t] + b_o) \tag{13}$$

$$Y_{vanilla\_t} = o_t \cdot tanh(Cell_t) \tag{14}$$

此处我们设置前序信息包含数量为 2。句首单词往往有着关键性作用, 上述设置能够最大程度上保留开头两个单词的原有信息; 同时, 确保真实反映其后的文本与前文本的关系。前序信息包含数量越大, 对其后文本的前序解码信息的负向加成越大。如果当前计算位置t小于或等于 2 时, 解码操作将仅使用朴素 LSTM 作为计算单元, 即最终解码输出Y_t等同于朴素 LSTM 输出$Y_{\text{vanilla_t}}$。若t大于 2 时, 解码操作将使用完整 PI-LSTM 作为计算单元, 即对朴素 LSTM 输出$Y_{\text{vanilla_t}}$进行二次更新。更新步骤如下:

在获得朴素 LSTM 输出$Y_{\text{vanilla_t}}$的同时, 解码器将前序解码输出Y_{t-2}和上下文向量Context$_t$送入 LSTM 单元计算出前序补充信息P_t。最后通过公式(16)使用权重W_{pre}对朴素 LSTM 输出$Y_{\text{vanilla_t}}$进行二次更新。

$$P_t = LSTM(Y_{t-2}, Context_t) \tag{15}$$

$$Y_t = (1 - W_{pre}) \cdot P_t + W_{pre} \cdot Y_{vanilla\_t} \tag{16}$$

PI-LSTM 将更多的前序解码信息集成到当前的解码操作中, 使得最终输出值Y_t真实反映出当前单词与先前文本的紧密关系。前序补充信息P_t可以纠正错误的语法结构, 并在预测语法树生成中发挥积极作用。

3 实验结果分析

3.1 数据集

为了与其他模型进行比较, 我们使用 HS(Hearthstone 炉石传说)[1]和 Django[20]数据集, 如表 1 所示。

表 1 数据集 DJANGO 和 HS 的输入序列

Table 1 The Input Sequence of DJANGO and HS

DJANGO	HS
get translation function attribute of the object t, call the result with an argument eol message, substitute the result for result.	*Maexxna 6 2 8 Destroy any minion damaged by this minion. Legendary*

HS 和 Django 是代码生成模型中常用的数据集。HS 数据集是实现纸牌游戏的 Python 类的集合。它的输入是关于卡牌的半结构化描述, 例如卡名, 描述, 费用等属性, 序列长度相对较短。Django 数据集是一个关于 Django Web 框架的代码行, 每行代码都有自然语言描述的注释。相比 HS, Django 数据集具有更长的输入语句且涵盖更多实际用例。

3.2 评估标准

在评估标准方面, 我们选择了准确率、双语评估研究(BLEU)和训练时间。准确率是生成语句与目标语句中相同单词数在句子长度中的占比。由于代码生成是相对复杂的问题, 仅靠准确率不可体现出模型的优劣性, 所以我们同时采用了双语评估研究(BLEU)指标。它常用于机器翻译领域, 以衡量生成文本与目标文本的相似性。相比准确率来说, BLEU 更具有说服力。在本实验中, 我们使用 bleu-4[24]作为 BLEU 指标, 它被广泛用于大多数以前的研究, 例如 SMT 模型[2], ASN 模型[8]和 CNN 模型[9]。此外, 我们使用训练时间作为评估指标的一部分, 其单位时间是分钟。

3.3 参数设置

我们的实验是在 GeForce 1070-ti 显卡和 Ubuntu 18.04 LST 系统上进行的。在优化器部分, 我们使用 Adam 优化器[25]。在注意力机制部分, 我们将可选注意力 K 的选定值预设为 14。在 Django 数据集上, epoch 为 50; 在 HS 数据集上, epoch 为 200。dropout 为 0.2。编码器 GRU 与解码器 PI-LSTM 隐藏层数均为 128。

3.4 对比实验

实验结果显示在表 2 中。我们在同一数据集上选择十一个经典模型进行比较。这十一个模型分别是: (1)

基于 Seq2Seq 框架的神经机器翻译模型(NMT)模型[4]; (2)生成多层树代码逻辑形式的 Seq2Tree 模型及其改进(3)Seq2Tree-UNK[5]; (4)使用指针网络的潜伏预测器网络模型(LPN)[1]; (5)基于语法的 Seq2Seq 神经网络模型(SNM)[7]; (6)抽象语法网络(ASN)模型及其改进(7)ASN+SUPATT[8]; (8)基于语义分析和代码生成的抽象解析器 TRANX[9]; (9)基于 Transformer[10]架构并优化捕获长依赖性语句的 TreeGen[11]模型及其(10)Self Attention 消融模型; (11)基于语法结构的 CNN 解码器模型[12]。

表 2　本文模型与 11 个经典模型的对比实验结果

Table 2　Results of Comparison Experiment: Performance of Our Model and the other ten Models

数据集	Django			HS		
模型	双语评估标准	准确率(%)	训练时间(min)	双语评估标准	准确率(%)	训练时间(min)
NMT	63.4	41.5%	–	60.4	1.5%	–
SEQ2TREE	44.6	28.9%	–	53.4	1.5%	–
SEQ2TREE-UNK	58.2	39.4%	–	62.8	13.6%	–
LPN	77.6	62.3%	–	67.1	6.1%	–
SNM	84.5	71.6%	341	75.8	16.2%	376
ASN	–	–	–	77.6	18.2%	–
ASN+SUPATT	–	–	–	79.2	22.7%	–
TRANX	–	73.7%	–	–	–	–
TREEGEN-Structural	–	–	–	80.8	**33.3%**	–
TREEGEN-SelfAttention	–	–	–	81.0	28.8%	–
CNN	–	–	–	79.6	27.3%	–
Our Model	**87.4**	**75.7%**	**297**	**87.5**	27.3%	**198**

根据表 2 可以看到, 我们的模型在 HS 数据集的准确率和 BLEU 得分方面优于其他模型。与 SNM 模型相比, 我们模型的准确率提升了 11.1%。就 BLEU 分数而言, 我们的模型相比 SNM 模型提升了 11.7, 并且比其他模型提升了至少 7.9。相比基于 Transformer 的 TreeGen 模型, 本文模型确实在准确率上稍逊一筹, 但我们的双语评估标准分数比 TreeGen 高出接近 7 分。在 Django 数据集上我们选取了五个模型, 实验结果出现了类似 HS 数据集的情况。我们的模型在此实验中也显示出最佳效果, 比 SNM 模型的准确率提高了 4.1%, BLEU 分数提高了 2.9, 且比解析器 TRANX 模型在 BLEU 分数提高了 2 分。该结果证明, 选择性局部注意力和 PI-LSTM 在提高准确率和 BLEU 中起着关键作用。与 SNM 模型所使用的软注意力相比, 选择性局部注意力有效地提高了 HS 数据集上某些长句子的上下文的紧密度, 使 BLEU 值得进一步改善。PI-LSTM 通过对输出的二次更新提高了基于 SNM 模型的准确率。当使用 PI-LSTM 作为解码器单元时, 与仅使用 LSTM 相比, 当前计算向量包含更多的前导信息。它加强了单词之间的联系, 尤其是在 Django 数据集上的长序列中, 从而大大提高了准确率和 BLEU。

在训练时间方面, 由于服务器的限制, 我们缺少除了 SNM 模型以外其他模型的结果。在 HS 数据集上, 本文模型的训练时间为 198 分钟, 在 Django 数据集上的训练时间为 297 分钟。与 SNM 模型的训练时间 376 分钟相比, 本文模型仅花费了 SNM 模型训练时间的一半; 与 SNM 模型的 341 分钟相比, 我们的模型仅使用了 87%的 SNM 训练时间。这证明了将 GRU 设置为编码运算单元可以大大减少训练时间。GRU 单元简化了 LSTM 中的门控运算, 减少了计算时间并节省了算力, 且对精度的负面影响很小。

3.5 消融实验

为了评估我们提出的模块的效果, 我们对模型进行了消融实验。下面将根据模型的结构分别进行框架消融实验和注意力消融实验, 以验证模型各部分的必要性。

3.5.1 框架消融实验

在本实验中, 我们替换了编码器和解码器的计算单元。框架消融测试用于验证 GRU 的有效性。框架消融实验结果如表 3 所示。

表 3　消融实验结果

Table 3　Results of Ablation Test

数据集		Django			HS	
框架消融实验	双语评估标准	准确率(%)	训练时间(min)	双语评估标准	准确率(%)	训练时间(min)
BiLSTM-LSTM	84.5	71.6%	341	75.8	16.2%	376
BiLSTM-PI-LSTM	80.6	73.7%	322	**83.8**	**21.2%**	352
GRU-PI-LSTM	**86.8**	**74.3%**	**299**	81.9	**21.2%**	**198**
注意力消融实验	双语评估标准	准确率(%)	训练时间(min)	双语评估标准	准确率(%)	训练时间(min)
GRU-PI-LSTM-Soft	86.8	74.3%	299	81.9	21.2%	**198**
Our Model	**87.4**	**75.7%**	**297**	**87.5**	**27.3%**	**198**

我们尝试用 BiLSTM 替换编码器单元, 并用朴素 LSTM 替换解码器单元, 并将注意力机制(此处使用软注意力)视为无关变量。鉴于 GRU 在绝大多数情况下仅减少了计算时间, 对准确率影响并不大, 故对 GRU-LSTM 进行实验并无意义。框架消融实验中的三个模型分别为 BiLSTM-LSTM、BiLSTN-PI-LSTM 和 GRU-PI-LSTM。

与 BiLSTM-LSTM 相比, GRU-PI-LSTM 表现出更好的效果。在 HS 和 DJANGO 数据集上, 准确性提高 2.7%~5%, BLEU 得分提高 1.3%~1.5%。除了 GRU-PI-LSTM 的 BLEU 低于 BiLSTN-PI-LSTM 之外, 其他评估标准也得到了改进。 HS 数据集的训练时间减少了 178 分钟, 仅为 BiLSTM-LSTM 所用时间的一半, 而 Django 数据集则减少了 42 分钟。

根据实验结果的比较, 我们可以认为, 与 BiLSTM 相比, GRU 简化了计算过程, 将训练时间缩短了将近 50%。在结合使用包含前序信息的长短期记忆网络 PI-LSTM 后, GRU 对准确率的负面影响可被忽略。PI-LSTM 使更多的前序解码信息参与解码, 从而提高了准确率和 BLEU 得分。GRU 和 PI-LSTM 的组合显示出相对较好的效果。

3.5.2 注意力消融实验

在此实验中, 我们更改了注意力机制。注意力消融实验用于验证选择性局部注意力对提高准确率和 BLEU 的积极影响。我们以本文模型为基础, 在注意力部分消融模型中将选择性局部注意力替换为软注意力, 该模型称为 GRU-PI-LSTM-SOFT, 实际上等同于框架消融实验中的 GRU-PI-LSTM。注意力消融实验结果如表 3 所示。

从结果可以看出, 与 GRU-PI-LSTM-SOFT 相比, 我们的模型显示出更好的效果。在 Django 数据集上, 本文模型的准确率提高了 1.4%; 在 HS 数据集上, 准确率提高了 6.1%。同时, 在 Django 数据集上, BLEU 分数提高了 0.6, 在 HS 数据集上, BLEU 分数提高了 5.6。注意力消融测试表明选择性局部注意力的 BLEU 得分均超过 85, 有助于提高生成代码的准确率, 还能使 HS 数据集上的长句子部分得到更好的理解和输出。注意力消融实验中的训练时间相似, 证实了注意力机制不对训练时间产生影响。

4　结语

本文提出了一种基于语法的代码生成模型。该模型基于使用 Seq2Seq 框架, GRU 编码器和 PI-LSTM 解码器的语法结构, 解决了在解码中未充分包含预定信息的问题, 并纠正了注意力机制的偏重, 更适合于训练大规模预处理数据。与其他十一个模型和三个消融模型相比, 我们的模型在准确率, BLEU 评分和训练时间方面具有很大的优势。本模型有望实现自动编程、在线代码纠正等需求任务以及可用于智能交互式设备自更新、AI 机器人等现实场景, 具有实在的现实意义。在未来的工作中, 我们将尝试使用双向 GRU 作为编码器计算单元, 希望可以通过紧密编码, 再次提高模型的准确率和 BLEU 分数。

致谢　研究工作得到华南师范大学计算机学院朱佳教授、汤庸教授以及广州市大数据智能教育重点实验室的支持, 表示衷心感谢。

参考文献

[1] Ling W, Grefenstette E, Hermann K M, et al. Latent predictor networks for code generation [EB/OL]. (2016–06–08)[2020–05–04]. https://arxiv.org/abs/1603.06744

[2] Brown P F, Cocke J, Della Pietra S A, et al. A statistical approach to machine translation. Computational Linguistics, 1990, 16(2): 79–85

[3] Sutskever I, Vinyals O, Le Q V. Sequence to sequence learning with neural networks // Advances in neural information processing systems. Cambridge, MA, 2014: 3104–3112

[4] Mandal S, Naskar S K. Natural language programing with automatic code generation towards solving addition-subtraction word problems // Proceedings of the 14th International Conference on Natural Language Processing (ICON-2017). Venice, 2017: 146–154

[5] Hochreiter S, Schmidhuber J. Long short-term memory. Neural computation, 1997, 9(8): 1735–1780

[6] Dong L, Lapata M. Language to logical form with neural attention [EB/OL]. (2016–06–06)[2020–05–04]. https://arxiv.org/abs/1601.01280

[7] Yin P, Neubig G. A syntactic neural model for general-purpose code generation [EB/OL]. (2017–04–06)[2020–05–04]. https://arxiv.org/abs/1704.01696

[8] Rabinovich M, Stern M, Klein D. Abstract syntax networks for code generation and semantic parsing [EB/OL]. (2017–04–25)[2020–05–04]. https://arxiv.org/abs/1704.07535

[9] Yin P, Neubig G. TRANX: a transition-based neural abstract syntax parser for semantic parsing and code generation [EB/OL]. (2018–10–05)[2020–05–04]. https://arxiv.org/abs/1810.02720v1

[10] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need // Advances in neural information processing systems. Red Hook, NY, 2017: 5998–6008

[11] Sun Z, Zhu Q, Xiong Y, et al. TreeGen: a tree-based transformer architecture for code generation // Association for the Advancement of Artificial Intelligence. New York, 2020: 8984–8991

[12] Sun Z, Zhu Q, Mou L, et al. A grammar-based structural CNN decoder for code generation // Proceedings of the AAAI Conference on Artificial Intelligence. New York, 2019, 33: 7055–7062

[13] Chung J, Gulcehre C, Cho K H, et al. Empirical evaluation of gated recurrent neural networks on sequence modeling [EB/OL]. (2014–12–11)[2020–05–04]. https://arxiv.org/abs/1412.3555

[14] Cho K, Van Merriënboer B, Gulcehre C, et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation [EB/OL]. (2014–09–03)[2020–05–04]. https://arxiv.org/abs/1406.1078

[15] Schuster M, Paliwal K K. Bidirectional recurrent neural networks. IEEE transactions on Signal Processing, 1997, 45(11): 2673–2681

[16] Joshi A K, Levy L S, Takahashi M. Tree adjunct grammars. Journal of computer and system sciences, 1975, 10(1): 136–163

[17] Schwarz C. Automatic syntactic analysis of free text. Journal of the American Society for Information Science, 1990, 41(6): 408–417

[18] Saul L K, Jaakkola T, Jordan M I. Mean field theory for sigmoid belief networks. Journal of artificial intelligence research, 1996, 4: 61–76

[19] Mikolov T, Kombrink S, Burget L, et al. Extensions of recurrent neural network language model // 2011 IEEE international conference on acoustics, speech and signal processing (ICASSP). Prague, 2011: 5528–5531

[20] Xu K, Ba J, Kiros R, et al. Show, attend and tell: Neural image caption generation with visual attention // International conference on machine learning. Lille, 2015: 2048–2057

[21] Luong M T, Pham H, Manning C D. Effective approaches to attention-based neural machine translation [EB/OL]. (2015–09–20)[2020–05–04]. https://arxiv.org/abs/1508.04025

[22] Oda Y, Fudaba H, Neubig G, et al. Learning to generate pseudo-code from source code using statistical machine translation // 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE). Lincoln, NE, 2015: 574–584

[23] Papineni K, Roukos S, Ward T, et al. BLEU: a method for automatic evaluation of machine translation // Proceedings of the 40th annual meeting of the Association for Computational Linguistics. Philadelphia, PA, 2002: 311–318

[24] Kingma D P, Ba J. Adam: A method for stochastic optimization [EB/OL]. (2017–01–30)[2020–05–04]. https://arxiv.org/abs/1412.6980

[25] Ortega-Bueno R, Rosso P, Pagola J E M. UO_UPV$_2$ at HAHA 2019: BiGRU neural network informed with linguistic features for humor recognition // Proceedings of the Iberian Languages Evaluation Forum (IberLEF'19), Co-Located with the 35th Conference of the Spanish Society for Natural Language Processing (SEPLN'19). Bilbao, 2019: 212–221