·人工智能与模式识别· 文章编号：1000-3428(2021)04-0092-08 文献标志码：A 中图分类号：TP312

一种用于代码注释自动生成的语法辅助复制机制

许柏炎，蔡瑞初，梁智豪
（广东工业大学 计算机学院，广州 510006）

摘 要：现有代码注释生成方法的复制机制未考虑源代码复杂多变的语法结构，导致存在准确率和鲁棒性不高等问题。通过改进指针网络使其支持结构化数据输入，提出一种语法辅助复制机制，以用于代码注释自动生成。该机制包含节点筛选策略和去冗余生成策略2个部分。节点筛选策略基于语法信息引入掩盖变量以过滤无效节点，从而降低指针网络对复杂语法的学习成本。去冗余生成策略基于时间窗口对节点概率进行动态调整，可解决代码自动注释中关键信息缺失的问题。实验结果表明，在WikiSQL数据集上，相比基准方法，该机制的BLEU、ROUGE-2和ROUGE-L指标值分别提升14.5%、10.3%和5.5%，在ATIS数据集上，上述指标值分别提升2.8%、6.6%和2.5%，验证了该机制的有效性以及引入语法信息的必要性。

关键词：代码注释生成；指针网络；自然语言生成；结构信息；复制机制

开放科学（资源服务）标志码（OSID）：

中文引用格式：许柏炎，蔡瑞初，梁智豪.一种用于代码注释自动生成的语法辅助复制机制[J].计算机工程，2021，47(4)：92-99.

英文引用格式：XU Boyan，CAI Ruichu，LIANG Zhihao.A grammar-aided copy mechanism for automatic code comment generation[J].Computer Engineering，2021，47(4)：92-99.

A Grammar-Aided Copy Mechanism for Automatic Code Comment Generation

XU Boyan，CAI Ruichu，LIANG Zhihao
（School of Computers，Guangdong University of Technology，Guangzhou 510006，China）

【Abstract】The copy mechanisms of the existing code comment generation methods do not consider the complex and varying grammar structures of source code，resulting in low copy accuracy and low robustness.This paper reconstructs the pointer network to make it support structured data input，and proposes a new grammar-aided copy mechanism for automatic comment generation.The mechanism consists of two parts：node filtering strategy and de-redundant generation strategy.Node filtering strategy that introduces masking variables to filter invalid type nodes based on grammatical information，which reduces the learning cost of complex grammar in pointer networks.De-redundant generation strategy that dynamically adjusts the node probability based on the time window，which solves the problem of missing key information in the automatically generated comment.Experimental results show that compared with baseline methods，the proposed method improves BLEU by 14.5%，ROUGE-2 by 10.3% and ROUGE-L by 5.5% on the WikiSQL dataset，and improves BLEU by 2.8%，ROUGE-2 by 6.6% and ROUGE-L by 2.5% on the ATIS dataset.The results verify the effectiveness of the mechanism and the necessity of introducing grammatical information.

【Key words】code comment generation；pointer network；natural language generation；structured information；copy mechanism
DOI：10. 19678/j. issn. 1000-3428. 0057290

0 概述

目前，代码托管平台Github的提交量已达10亿，涉及337种编程语言，但是其中大量代码缺乏可读的自然语言注释[1]。代码注释是软件开发的重要环节，可大幅提高代码的可读性，但是其过程十分耗时[2]。随着深度学习技术的应用，自然语言生成任务得到快速发展，其中典型代表有机器翻译[3-4]和文本摘要生成[5-7]等任务。代码自动注释是自然语言生成中最具挑战性的任务之一，目标是基于输入代码

基金项目：国家自然科学基金（61876043）；广东省自然科学基金（2014A030306004，2014A030308008）；广东特支计划（2015TQ01X140）；NSFC-广东联合基金（U1501254）；广州市珠江科技新星专项（201610010101）；广州市科技计划项目（201902010058）。
作者简介：许柏炎（1991—），男，博士研究生，主研方向为自然语言处理；蔡瑞初（通信作者），教授、博士生导师；梁智豪，硕士研究生。
收稿日期：2020-01-27 **修回日期：**2020-04-11 **E-mail：**cairuichu@gmail.com

生成相应的可理解的自然语言注释或描述。

早期的研究人员将代码注释生成建模为源代码序列到自然语言注释序列的翻译任务，但是不足之处在于其仅提取源代码中的文本信息而忽略了结构信息。近年来，学者们考虑引入源代码的结构化输入，如抽象语法树（Abstract Syntax Tree，AST），并采用机器学习的方法或模型来提取源代码中结构信息的语义表达。然而，这些研究工作未充分考虑如何从结构化数据输入中复制提取关键信息。代码注释生成的目标注释中通常会出现目标词典中不存在的单词（Out Of Vocabulary，OOV），如数值型、字符型等变量单词，从而导致生成的自然语言注释输出不完整。为解决该问题，研究人员在代码注释生成模型中引入指针网络[8]，指针网络通过在单位时间步输出指向输入序列的某一个单词，从而形成一种复制机制。但是，经典的指针网络只支持代码的序列化输入，不支持代码的结构化输入，从而难以利用源代码的结构化信息。同时，源代码的抽象语法树存在复杂多变的语法结构，使得指针网络需花费大量的学习成本来学习复杂语法结构，导致难以准确地提取核心信息。此外，引入复制机制的代码注释生成模型倾向于频繁复制核心词，使得生成的自然语言注释较易出现冗余信息。

本文提出一种用于代码注释自动生成的语法辅助复制机制。将编码器-解码器作为基本框架，编码器为树型长短期记忆网络（Tree-LSTM）[9]，解码器为长短期记忆网络（LSTM）[10]。由于经典的指针网络仅支持序列结构网络，无法直接应用于上述框架，因此本文通过改进指针网络结构使其支持源代码的抽象语法树输入。具体地，在编码阶段，编码器网络通过对源代码的抽象语法树进行编码得到结构化的语义信息并存储于节点的隐状态中。在解码阶段，解码器网络在每个时间步通过当前隐状态和输入所有节点的隐状态来计算注意力向量，并输入到二元分类器中以选择复制机制或生成机制。本文所提语法辅助复制机制包含基于语法信息的节点筛选策略和基于时间窗口的去冗余生成策略，前者根据语法节点类型使用掩码向量过滤无关的语法节点，后者在时间窗口内对被复制过的节点进行概率惩罚，从而抑制重复复制该节点的现象。

1　相关工作

多数代码注释生成任务主要关注源代码的文本信息。具体地，文献[11]通过主题模型和n-gram技术来生成代码注释，文献[12]提出一种基于长短时记忆网络与注意力机制的语言模型Code-NN，从而生成C#和SQL注释，文献[13]利用卷积神经网络与注意力机制来预测生成代码注释。

近年来，研究人员除考虑源代码的文本信息外，还探究如何提取其抽象语法树中的结构信息。文献[14]提出结合语法结构信息和文本信息的语义表达方法。文献[15]通过遍历Java的语义解析树节点获得序列形式，将其作为输入以捕获语义特征和语法信息。文献[16]提出的Graph2Seq将结构化查询语言（SQL）序列看作有向图，利用图网络学习全局结构信息。本文同样采用树型结构编码器，输入源代码的语义解析树以提取结构化信息。

目前，研究人员普遍采用经典的复制机制解决自然语言注释生成的OOV问题，如文献[17]提出结合复制机制的混合生成网络以进行文本摘要生成，文献[18]利用复制机制复制子句，文献[19]利用复制机制进行代码生成。与上述工作不同，本文建立一种支持树型结构化数据输入的指针网络变体，并基于源代码的语法结构和特性，提出包含节点筛选策略和去冗余生成策略的语法辅助复制机制，以从源代码的语义解析树中提取关键内容信息。

2　代码自动注释模型

代码自动注释模型的输入一般是源代码对应的抽象语法树表示，而非源代码本身[20]。抽象语法树省略编程语法上的部分无用细节，只保留语法结构以及与代码内容相关的部分，作为源代码语法结构的抽象化表示。以SQL查询语句为例，其对应的抽象语法树表示如图1所示。

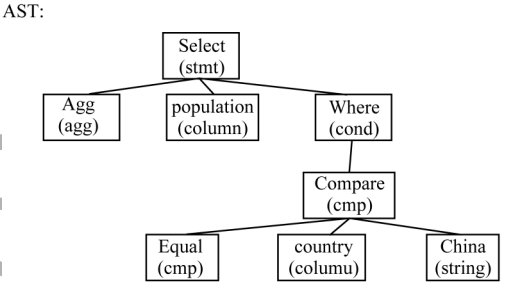

图1　SQL查询语句对应的抽象语法树表示

Fig.1　Abstract syntax tree representation of SQL query statement

本文对代码自动注释任务进行定义：给定一个源代码的抽象语法树 $X=\{x_1, x_2, \cdots, x_{|X|}\}$，目标的自然语言注释为 $Y=\{y_1, y_2, \cdots, y_{|Y|}\}$，其中，$x_i$ 代表抽象语法树的第 i 个节点，y_j 是自然语言注释序列的第 j 个单词。本文建立的代码自动注释模型是基于编码器-解码器框架设计的，结合编码器（树型长短期记忆网络[9]）、解码器（长短期记忆网络[10]）和注意力机制[21]。在编码阶段，树型长短期记忆网络自下而上递归式地遍历抽象语法树，最终以根节点的隐含层表示作为抽象语法树的语义表达向量；在解码阶段，长短期记忆网络接收编码器的语义表达并解码出目标序列，在每个时间步基于当前的隐状态与编码阶段得到的抽象语法树上节

点的隐状态，通过注意力机制[21]计算注意力向量以复制或生成目标单词。本文模型在给定代码抽象语法树X下，生成对应的自然语言注释序列Y。其中，每个时间步t的预测输出为y_t，条件概率为p。本文采用极大似然估计(Maximum Likelihood Estimation，MLE)优化如下的目标函数：

$$p(Y|X)=\prod_{t=1}^{|Y|}p(y_t|y_{<t},X) \tag{1}$$

为了解决OOV问题，本文将复制机制引入到代码注释生成模型中。复制机制使得生成模型除了从目标词典中选择单词外，还可从源代码输入中提取信息。具备复制机制的代码注释生成模型的目标函数可基于式(1)展开为：

$$\begin{aligned}p(Y|X)&=\prod_{t=1}^{|Y|}p(y_t|y_{<t},X)=\\&\prod_{t=1}^{|Y|}[p(y_t|a_t=\text{gen},y_{<t},X)p(a_t=\text{gen}|y_{<t},X)+\\&p(y_t|a_t=\text{copy},y_{<t},X)p(a_t=\text{copy}|y_{<t},X)]\end{aligned} \tag{2}$$

其中，a_t表示第t个时间步执行的解码动作，可为输入复制或词典生成，即$a_t\in\{\text{gen},\text{copy}\}$，$p(a_t|y_{<t},X)$表示一个伯努利分布，且$p(a_t|y_{<t},X)\in\{0,1\}$。

为了从源代码的复杂语法结构中提取信息，本文提出一种语法辅助复制机制，该机制包含基于语法信息的节点筛选策略与基于时间窗口的去冗余生成策略。如图2所示，编码器(Encoder)对源代码的抽象语法树进行编码，将编码后的语义表达(enc.embed)传递到解码器(Decoder)进行解码。解码器使用LSTM在每个时间步进行从源输入复制(CP)或从词典生成(GEN)操作。解码器若选择CP操作，将执行基于语法信息的节点筛选策略(①)和基于时间窗口的去冗余生成策略(②)；否则，解码器选择GEN操作，直接从词典中选择单词。长虚线区域内的节点为经过节点筛选策略筛选后的候选语法节点，短虚线框为在时间步t时“China”节点的被复制概率的衰减过程，γ表示衰减持有率，λ_{t-1}表示上一个时间步的衰减率，p为每个节点具体的复制概率。节点筛选策略过滤抽象语法树的无关节点，保留字符型变量节点；去冗余生成策略对已被复制的节点“China(string)”进行惩罚，防止该节点被错误地重复复制。

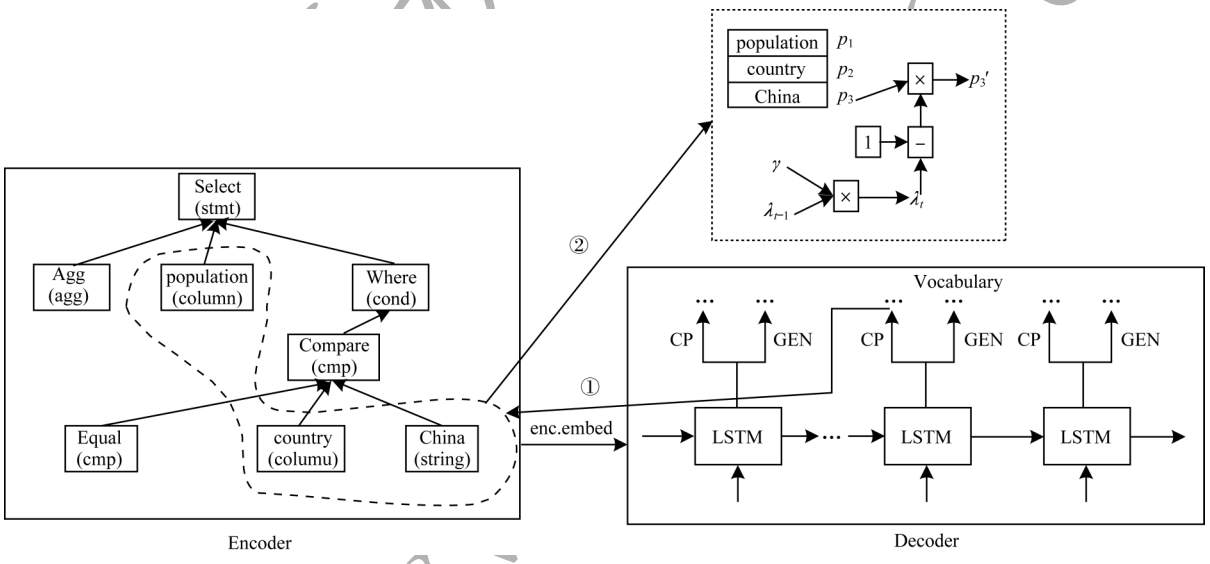

图2 代码自动注释模型整体架构

Fig.2 Overall architecture of automatic code comment model

3 抽象语法树编码

编码阶段的目标是得到抽象语法树的语义编码表达，本节将介绍公式化编码过程。编码阶段分为语法节点向量化和抽象语法树向量化2个过程。

3.1 语法节点向量化

抽象语法树X由多个节点组成，每个节点x_i可能包含一个或者多个单词，即$x_i=\left\{\omega_{i1},\omega_{i2},\cdots,\omega_{i|x_i|}\right\}$。初始化词典词嵌入矩阵为$\boldsymbol{W}_e\in\boldsymbol{R}^{l_e\times d}$，其中，$l_e$为输入词典的大小，$d$为词向量维度大小。对节点$i$中的第$j$个单词$\omega_{ij}$构建词典的独热(one-hot)索引变量$\text{id}(\omega_{ij})\in\boldsymbol{R}^{l_e}$，可以得到$\omega_{ij}$的词嵌入向量如下：

$$\boldsymbol{emb}^{(e)}(\omega_{ij})=\boldsymbol{W}_e^{\mathrm{T}}\times\text{id}(\omega_{ij}) \tag{3}$$

通过式(3)可以得到节点中每个单词的词嵌入向量，为了得到节点的向量化表示，需要获得节点中的单词序列表达。本文采用LSTM[10]作为序列学习网络，在每个时间步循环地输入序列中每个单词ω_{ij}的词嵌入向量，并存储更新隐含层信息，如下：

$$\boldsymbol{h}_{ij}=\text{LSTM}(\boldsymbol{emb}^{e}(\omega_{ij}),\boldsymbol{h}_{ij-1}) \tag{4}$$

因此，最后一个时间步的隐含层状态$\boldsymbol{h}_{i|x_i|}\in\boldsymbol{R}^d$包含该序列的语义信息，可作为节点$x_i$的向量化表

示 $\boldsymbol{v}_i \in \boldsymbol{R}^d$，如下：

$$\boldsymbol{v}_i = \boldsymbol{h}_{i|x_i|} \tag{5}$$

3.2　抽象语法树向量化

通过3.1节获得抽象语法树的节点向量化，编码器需进一步将节点的表达通过抽象语法树的结构聚合成抽象语法树的整体表达。为了捕捉抽象语法树的语法结构，将Tree-LSTM[9]作为编码器，进一步对抽象语法树进行编码。Tree-LSTM自下而上递归式和结构化地输入抽象语法树数据，将信息更新至节点隐状态。与经典的LSTM网络相同，Tree-LSTM同样包含输入门、遗忘门和输出门，每个门都是基于节点的向量表示及其孩子节点的隐状态。当抽象语法树是N-ary树时，语法树的非叶子节点最多存在 N 个孩子节点。对于任意节点 j，其第 k 个孩子的隐状态和单元状态（Cell State）分别为 $\boldsymbol{s}_{jk}$ 和 $\boldsymbol{c}_{jk}$，则对节点 j 隐状态的更新过程如下：

$$\boldsymbol{i}_j = \sigma\left(\boldsymbol{W}^{(i)}\boldsymbol{v}_j + \sum_{l=1}^{N}\boldsymbol{U}_l^{(i)}\boldsymbol{s}_{jl} + \boldsymbol{b}^{(i)}\right) \tag{6}$$

$$\boldsymbol{f}_{jk} = \sigma\left(\boldsymbol{W}^{(f)}\boldsymbol{v}_j + \sum_{l=1}^{N}\boldsymbol{U}_{kl}^{(f)}\boldsymbol{s}_{jl} + \boldsymbol{b}^{(f)}\right) \tag{7}$$

$$\boldsymbol{o}_j = \sigma\left(\boldsymbol{W}^{(o)}\boldsymbol{v}_j + \sum_{l=1}^{N}\boldsymbol{U}_l^{(o)}\boldsymbol{s}_{jl} + \boldsymbol{b}^{(o)}\right) \tag{8}$$

$$\boldsymbol{u}_j = \tanh\left(\boldsymbol{W}^{(u)}\boldsymbol{v}_j + \sum_{l=1}^{N}\boldsymbol{U}_l^{(u)}\boldsymbol{s}_{jl} + \boldsymbol{b}^{(u)}\right) \tag{9}$$

$$\boldsymbol{c}_j = \boldsymbol{i}_j \odot \boldsymbol{u}_j + \sum_{l=1}^{N}\boldsymbol{f}_{jl} \odot \boldsymbol{c}_{jl} \tag{10}$$

$$\boldsymbol{s}_j = \boldsymbol{o}_j \odot \tanh\left(\boldsymbol{c}_j\right) \tag{11}$$

其中，$\boldsymbol{i}_j$、$\boldsymbol{f}_{jk}$、$\boldsymbol{u}_j$、$\boldsymbol{o}_j \in \boldsymbol{R}^d$ 分别代表输入门、遗忘门、输出门和被用来更新细胞状态 $\boldsymbol{c}_j \in \boldsymbol{R}^d$ 的状态向量，$\boldsymbol{W}^{(\cdot)} \in \boldsymbol{R}^{d\times d}$ 和 $\boldsymbol{U}^{(\cdot)} \in \boldsymbol{R}^{d\times d}$ 代表可学习的参数矩阵，$\boldsymbol{b}^{(\cdot)} \in \boldsymbol{R}^d$ 代表偏置向量，$\boldsymbol{v}_j \in \boldsymbol{R}^d$ 代表节点 j 的向量化表示。

Tree-LSTM编码器递归式地计算抽象语法树以更新各节点的隐含层信息，根节点的隐含层状态含有其他所有节点的语义结构信息，因此，本文将根节点的隐含层状态作为抽象语法树的语义结构表达，用于解码器更新。

4　自然语言注释

4.1　基于注意力的解码过程

模型的解码器负责将从编码器得到的抽象语法树结构语义表达迭代式地生成自然语言注释。本文解码器采用LSTM[10]，以结构语义表达作为初始化隐状态，按单位时间步更新输出新的隐状态，且解码器输出的隐状态用于构造注意力向量。具体地，对于任意时间步 t，解码器LSTM输出的隐状态为 $\boldsymbol{o}_t \in \boldsymbol{R}^d$，注意力向量 $\boldsymbol{q}_t \in \boldsymbol{R}^d$ 的计算过程如下：

$$\boldsymbol{r}_{tj} = \boldsymbol{s}_j \times \boldsymbol{o}_t \tag{12}$$

$$\boldsymbol{\alpha}_t = \mathrm{softmax}\left(\boldsymbol{r}_t\right) \tag{13}$$

$$\boldsymbol{z}_t = \sum_{j=1}^{n}\boldsymbol{\alpha}_{tj}\boldsymbol{s}_j \tag{14}$$

$$\boldsymbol{q}_t = \tanh\left(\boldsymbol{W}_{\mathrm{attn}}\left[\boldsymbol{z}_t;\ \boldsymbol{o}_t\right]\right) \tag{15}$$

从上述公式可知，在解码器阶段，根据当前时间步 t 的隐状态 $\boldsymbol{o}_t$ 与编码器得到的抽象语法树各节点的隐状态分别计算得分 $\boldsymbol{r}_{tj}$；接着，通过softmax函数进行归一化得到权重向量 $\boldsymbol{\alpha}_t \in \boldsymbol{R}^{|X|}$；然后，对各节点隐状态进行加权求和得到当前时间步的内容向量 $\boldsymbol{z}_t \in \boldsymbol{R}^d$；最终，利用可学习参数矩阵 $\boldsymbol{W}_{\mathrm{attn}} \in \boldsymbol{R}^{d\times 2d}$ 和tanh激活函数融合内容向量 $\boldsymbol{z}_t$ 和当前解码隐状态 $\boldsymbol{o}_t$，得到注意力向量 $\boldsymbol{q}_t \in \boldsymbol{R}^d$。通过上述注意力机制过程，解码器生成自然语言注释时可以更关注抽象语法树的相关节点，使得生成模型输出的自然语言注释具备更高的语义准确性。

为了解决自然语言生成的OOV问题，本文模型引入复制机制，使其具备从源输入中提取关键信息的能力。本文模型存在2种生成来源，分别是从目标词典中预测和从源输入中复制提取。因此，该模型需要一种机制来选择生成来源，本文构造基于注意力向量输入的分类器，以选择生成来源。构造参数为 $\boldsymbol{W}_{gc} \in \boldsymbol{R}^{2\times d}$ 的全连接神经网络，输入注意力向量产生 $a_t \in \{\mathrm{gen}, \mathrm{copy}\}$ 二元概率分布，如下：

$$p\left(a_t \mid y_{<t}, X\right) = \mathrm{softmax}\left(\boldsymbol{W}_{gc} \times \boldsymbol{q}_t\right) \tag{16}$$

当预测概率大于复制概率时，解码器选择从目标词典中生成。具体地，构造参数为 $\boldsymbol{W}_{\mathrm{gen}} \in \boldsymbol{R}^{l_d\times d}$ 的全连接神经网络，输入注意力向量产生目标词典的概率分布，如下：

$$p(y_t \mid a_t = \mathrm{gen}, y_{<t}, X) = \mathrm{softmax}(\boldsymbol{W}_{\mathrm{gen}} \times \boldsymbol{q}_t) \tag{17}$$

4.2　语法辅助复制机制

4.1节介绍了解码器生成来源为目标词典下的解码过程，本节将重点阐述当生成来源为源输入信息时的解码过程，并通过语法辅助来提高复制提取的准确率。

当生成来源分类器输出的复制概率大于预测概率时，解码器将从输入信息中进行复制提取。具体地，对时间步 t 的注意力向量 $\boldsymbol{q}_t$，计算其与编码器中抽象语法树各节点隐状态的得分向量 $\boldsymbol{g}_t \in \boldsymbol{R}^{|X|}$，归一化后选择概率最大的节点并复制其中的文本信息，过程如下：

$$\boldsymbol{g}_{tj} = \boldsymbol{s}_j \times \boldsymbol{q}_t \tag{18}$$

$$p\left(y_t \mid a_t = \mathrm{copy}, y_{<t}, X\right) = \mathrm{softmax}\left(\boldsymbol{g}_t\right) \tag{19}$$

从上述过程可以看出，复制机制与注意力机制实现过程相似，但是复制机制依赖于注意力机制的注意力向量并进行二次计算。

4.2.1　基于语法信息的节点筛选策略

代码自动注释任务的输入是程序代码的抽象语法树，其具有复杂多变的结构信息。现有的复制机制多为非结构化输入数据而设计，模型需要花费大量的时间学习如何从复杂结构中提取有用的信息，且提取效果通常不佳。本文利用语法辅助复制机制从抽象语法树中提取关键信息。对于代码自动注释任务而言，关键信息一般为抽象语法树上的数值型节点和字符型节点。因此，本文将复制过程的搜索范围从语法树所有类型的节点压缩至这两类节点，该过程可以降低搜索空间，也能够减少抽象语法树结构变化所带来的干扰以及神经网络的学习成本。

本文通过引入掩盖变量来过滤无效候选节点。假设时间步 t 的节点掩盖向量为 $\boldsymbol{d}_t \in \boldsymbol{R}^{|X|}$，长度为节点数，取值为0或 $-\infty$，0代表对应语法节点有效，$-\infty$ 代表节点无效，将其从候选节点集中剔除。在复制过程中，剔除无效节点后余下候选节点的概率计算公式如下：

$$p_{\text{mask}}\left(y_t \mid a_t=\text{copy}, y_{<t}, X\right)=\text{softmax}\left(\boldsymbol{g}_t+\boldsymbol{d}_t\right) \tag{20}$$

由上述过程可得到复制机制下当前时间步对应的损失函数为：

$$L_t=-\log_a p_{\text{mask}}\left(y_t \mid a_t=\text{copy}, y_{<t}, X\right) \tag{21}$$

一方面，通过掩盖向量可以剔除无效节点，使得无效节点集概率恒等于0，正确节点掩盖后的概率大于等于原始概率，当前时间步的损失下降，神经网络的整体成本函数也会下降；另一方面，从候选节点集合中过滤无效候选节点，可以使得当前时间步搜索空间得到一定的压缩。因此，节点筛选策略在一定程度上可以加速神经网络的学习。

4.2.2 基于时间窗口的去冗余生成策略

现有代码自动注释方法生成的自然语言注释存在主要关键词冗余与部分关键词缺失的现象，造成该现象的原因是复制机制与注意力机制原理相近，导致指针网络倾向于重复指向最具注意力的少数节点，而忽略了其他有用节点。为解决上述问题，本文提出一种基于时间窗口的去冗余生成策略，通过在时间窗口内对已被复制的语法节点的复制概率进行衰减，从而提高候选集合中其他有用且未被复制的节点被初次复制的概率。具体地，对于一个已被复制的语法节点 x_i，设置基于时间窗口变化的衰减率 λ_{it}，用超参数 $\gamma \in (0,1)$ 表示衰减持有率，用于控制每次衰减率的减少量。λ_{it} 表示如下：

$$\lambda_{it}=\begin{cases}0, x_i \text{未被复制}\\ 1, x_i \text{在}t\text{时刻被复制}\\ \gamma\lambda_{it-1}, x_i \text{在}t\text{时刻前被复制}\end{cases} \tag{22}$$

在时间步 t 下，抽象语法树候选节点集合 $X=\left\{x_1, x_2, \cdots, x_{|X|}\right\}$ 有一一对应的衰减变量集合 $\lambda_t=\left\{\lambda_{1t}, \lambda_{2t}, \cdots, \lambda_{|X|t}\right\}$。结合基于语法信息的节点筛选策略和基于时间窗口的去冗余生成策略，最终抽象语法树复制候选节点的概率分布如下：

$$p_{\text{final}}(y_t \mid a_t=\text{copy}, y_{<t}, X)=\text{softmax}(\boldsymbol{g}_t+\boldsymbol{d}_t)\times(1-\lambda_t) \tag{23}$$

5 实验结果与分析

5.1 数据集

本文采用结构化查询语言（SQL）和逻辑表达式（Lambda）2种不同编程语言的源代码数据进行测试，源代码数据分别来自WikiSQL[22]和ATIS[23]数据集。WikiSQL数据集数据来源于维基百科，含有24 241个小规模数据表格，基于以上数据表格产生80 654个结构化查询语句-自然语言对。ATIS数据集数据来源于现实航空业务数据库，含有25个真实数据表，基于以上数据表格产生5 373个逻辑表达-自然语言对。为了验证各方法的有效性，将WikiSQL数据集进一步划分为分别含有56 355个样本的训练集、8 421个样本的验证集和15 878个样本的测试集。同样地，将ATIS数据集划分为分别含4 434个样本的训练集、491个样本的验证集和448个样本的测试集。上述数据集涉及的2种编程语言均可通过抽象语法描述语言（Abstract Syntax Description Language，ASDL）来获得对应的抽象语法树。其中，SQL结构化查询语言和Lambda逻辑表达式的ASDL语法分别在文献[24]和文献[25]中有所定义。

5.2 基准方法

本文将具有代表性的两类结构性方法作为实验的基准方法，其中包括树至序列（Tree to Sequence，Tree2Seq）模型[26]和图至序列（Graph to Sequence，Graph2Seq）模型[16]，以上基准方法均采用注意力机制。需要注意的是，Tree2Seq模型输入结构为源代码的抽象语法树，而Graph2Seq模型的原文输入结构是专为SQL查询语句设计的图结构，其无法支持Lambda逻辑表达式，而且原文没有提供为SQL查询语句转换的预处理代码。因此，本文将统一采用抽象语法树作为Graph2Seq模型的输入结构。编程语言的抽象语法树表示具有普遍性，可应用于大部分编程语言。具体地，本文将源代码的抽象语法树看作一个无向图，父、子节点之间的连接看作图节点的边。此外，为了更好地验证语法辅助复制机制的性能，本文在Tree2Seq模型中引入一种传统的复制机制。

5.3 评价指标

本文采用BLEU[27]和ROUGE[28]作为自然语言注释生成的评价指标。BLEU是自然语言处理任务常用的评价指标，其通常从字符或单词级别来评价2个文本或2个文本集合的相似度。ROUGE指标评价所生成文本句子的充分性和忠实性，为了提高所生成文本的连贯性，本文使用考虑文本上下文的ROUGE指标，分别为ROUGE-2和ROUGE-L。

5.4 实验设置

实验参数设置如下：词向量、编码器和解码器的隐状态维度大小均为128维，批量大小均设置为32个，梯度优化器选择Adam优化器[29]，学习率设置为0.001。此外，为了降低词典大小，源词典和目标词典只记录词频不低于4的单词。实验没有采用由其他语料库预训练的词向量，如Word2Vec[30]和Glove[31]等，而是使用xavier初始化器[32]随机初始化词向量。本文所提复制机制中基于时间窗口的去冗余生成策略的衰减持有率，在WikiSQL和ATIS数据集中分别设置为80%和90%时系统性能达到最优，5.5.3节将对该参数进行详细的敏感性分析。

5.5 结果分析

5.5.1 模型性能分析实验

表1和表2所示分别为本文模型和基线模型在WikiSQL和ATIS 2个数据集上进行代码自动生成的实验结果，其中最优结果加粗表示。从中可以看出，本文模型在WikiSQL数据集中的性能表现明显优于2个基

线模型，与最佳基线模型Tree2Seq+copy相比，本文模型的BLEU提升14.5%，ROUGE-2和ROUGE-L分别提升10.3%和5.5%。在ATIS数据集上，与最佳基线模型Tree2Seq+copy相比，本文模型的BLEU提升2.8%，ROUGE-2和ROUG-L分别提升6.6%和2.5%。上述结果表明，本文模型相比现有代码注释生成方法有显著的性能提升，其能生成更加准确的代码自然语言注释。

表1　3种模型在WikiSQL数据集上的性能对比结果

Tabel 1　Performance comparison results of three models on WikiSQL dataset

模型	BLEU-4	ROUGE-2	ROUGE-L
Tree2Seq+copy模型	31.0	36.8	54.5
Graph2Seq模型	17.6	24.3	45.7
本文模型	**35.5**	**40.6**	**57.5**

表2　3种模型在ATIS数据集上的性能对比结果

Tabel 2　Performance comparison results of three models on ATIS dataset

模型	BLEU-4	ROUGE-2	ROUGE-L
Tree2Seq+copy模型	39.0	43.7	58.4
Graph2Seq模型	34.6	41.8	58.3
本文模型	**40.1**	**46.6**	**59.9**

5.5.2　节点筛选策略的有效性实验

为了量化节点筛选策略在压缩搜索空间方面的性能，本文统计WikiSQL和ATIS数据集单位时间步内的平均候选节点个数，结果如表3所示。从表3可以看出，通过节点筛选策略过滤无效的语法节点，在WikiSQL数据集中候选节点空间的压缩比例超过60%，在ATIS数据集中压缩比例超过50%，即节点筛选策略能大幅降低代码注释生成的搜索空间。

表3　2种数据集在单位时间步内的平均候选节点数

Table 3　The average number of candidate nodes per unit time step of two datasets

数据集	原始候选节点数	筛选后候选节点数
WikiSQL验证集	11.09	3.72
	11.14	3.74
	11.17	3.75
ATIS验证集	33.90	13.04
	34.01	13.05
	29.54	11.39

为了验证节点筛选策略在收敛速度方面的性能，本文在WikiSQL的验证集下测试BLEU指标的收敛趋势。为了公平有效地对比，在本文模型中去除去冗余生成策略，仅保留节点筛选策略，并与结合复制机制的Tree2Seq模型进行对比，结果如图3所示。从图3可以看出，相比基线模型，仅保留节点筛选策略的本文模型收敛速度有所提升，即节点筛选策略能在一定程度上加速神经网络的学习。

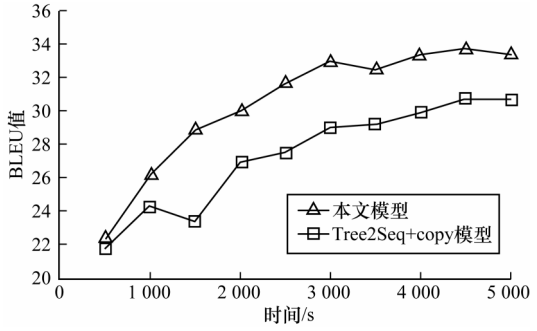

图3　2种模型在WikiSQL验证集上的收敛速度对比

Fig.3　Convergence rate comparison of two models on WikiSQL verification set

5.5.3　衰减持有率的敏感性实验

为了验证去冗余生成策略中的衰减持有率对模型影响的敏感程度，本文在WikiSQL和ATIS数据集上设计针对衰减持有率的敏感性实验，结果如图4和图5所示。从图4可以看出，在WikiSQL数据集上，当衰减持有率为80%时，模型达到该数据集上的最高BLEU值。从图5可以看出，在ATIS数据集上，当衰减持有率为90%时，模型达到该数据集上的最高BLEU值。衰减持有率能够直观反映自然语言注释中一个字或词重复出现的间距，图4和图5的实验结果表明，在2个数据集上重复字词出现的间距均较大。

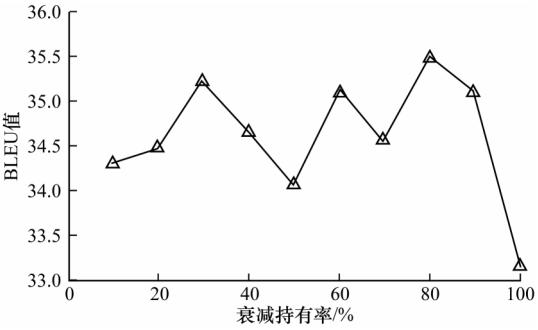

图4　WikiSQL测试集上的衰减持有率敏感性实验结果

Fig.4　Sensitivity experimental results of decay keeping probability on WikiSQL test set

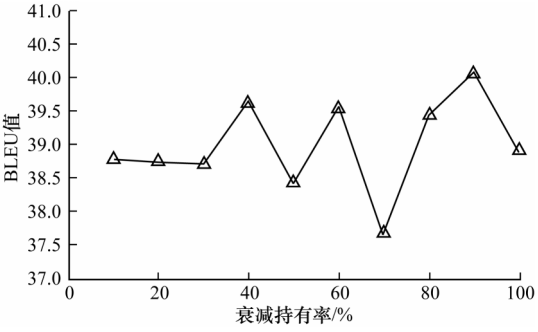

图5　ATIS测试集上的衰减持有率敏感性实验结果

Fig.5　Sensitivity experimental results of decay keeping probability on ATIS test set

5.5.4 分离实验

为了更好地体现基于语法信息的节点筛选策略和基于时间窗口的去冗余生成策略在模型中的重要性，本文设计分离实验以验证去除其中一种策略后的模型性能，测量不同的变种模型在WikiSQL和ATIS验证集上的BLEU值变化。本文定义如下的变种模型：

1）Ours-SA：只去除基于语法信息的节点筛选策略的模型，其他保持不变。

2）Ours-CD：只去除基于时间窗口的去冗余生成策略的模型，其他保持不变。

分离实验结果如表4所示，从表4可以看出，与原始模型相比，Ours-SA和Ours-CD的BLEU值在2个数据集上均出现明显下降，验证了本文所提2种策略的有效性。此外，基于时间窗口的去冗余生成策略对WikiSQL影响较大，去除该策略后模型的BLEU值降低超过10%，而2种策略对ATIS数据集的影响相差不大。

表4 2种数据集上的分离实验结果

Table 4 Results of separation experiments on two datasets

模型	WikiSQL数据集	ATIS数据集
Ours-SA模型	35.2	41.3
Ours-CD模型	31.8	41.1
本文模型	35.9	42.7

6 结束语

本文提出一种用于代码注释自动生成的语法辅助复制机制，其中主要包括节点筛选策略和去冗余生成策略2个部分。实验结果表明，与Tree2Seq+copy和Graph2Seq模型相比，该机制的BLEU和ROUGE指标值均较高。下一步将在代码注释生成模型框架中引入语法信息，以提升模型对复杂语法的学习效果。

参考文献

[1] BEN-NUN T, JAKOBOVITS A S, HOEFLER T. Neural code comprehension: a learnable representation of code semantics[C]//Proceedings of the 32nd International Conference on Neural Information Processing Systems. New York, USA: ACM Press, 2018: 3585-3597.

[2] ALLAMANIS M, TARLOW D, GORDON A, et al. Bimodal modelling of source code and natural language [C]// Proceedings of the 32nd International Conference on Machine Learning. New York, USA: ACM Press, 2015: 2123-2132.

[3] LIU Yang. Recent advances in neural machine translation[J]. Journal of Computer Research and Development, 2017, 54(6): 1144-1149. (in Chinese)
刘洋. 神经机器翻译前沿进展[J]. 计算机研究与发展, 2017, 54(6): 1144-1149.

[4] SUTSKEVER I, VINYALS O, LE Q V. Sequence to sequence learning with neural networks[C]//Proceedings of the 27th International Conference on Neural Information Processing Systems. New York, USA: ACM Press, 2014: 3104-3112.

[5] WU Renshou, WANG Hongling, WANG Zhongqing, et al. Short text summary generation with global self-matching mechanism[J]. Journal of Software, 2019, 30(9): 2705-2717. (in Chinese)
吴仁守, 王红玲, 王中卿, 等. 全局自匹配机制的短文本摘要生成方法[J]. 软件学报, 2019, 30(9): 2705-2717.

[6] MING Tuosiyu, CHEN Hongchang, HUANG Ruiyang, et al. Semantic subgraph predictive summary algorithm based on weighted AMR graph[J]. Computer Engineering, 2018, 44(10): 292-297, 302. (in Chinese)
明拓思宇, 陈鸿昶, 黄瑞阳, 等. 基于加权AMR图的语义子图预测摘要算法[J]. 计算机工程, 2018, 44(10): 292-297, 302.

[7] GAMBHIR M, GUPTA V. Recent automatic text summarization techniques: a survey[J]. Artificial Intelligence Review, 2017, 47(1): 1-66.

[8] VINYALS O, FORTUNATO M, JAITLY N. Pointer networks[EB/OL]. [2019-12-20]. https://arxiv.org/pdf/1506.03134.pdf.

[9] TAI K S, SOCHER R, MANNING C D. Improved semantic representations from tree-structured long short-term memory networks[C]//Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing. New York, USA: ACM Press, 2015: 1556-1566.

[10] HOCHREITER S, SCHMIDHUBER J. Long short-term memory[J]. Neural Computation, 1997, 9(8): 1735-1780.

[11] MOVSHOVITZ-ATTIAS D, COHEN W. Natural language models for predicting programming comments [C]// Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics. New York, USA: ACM Press, 2013: 35-40.

[12] IYER S, KONSTAS I, CHEUNG A, et al. Summarizing source code using a neural attention model[C]//Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics. New York, USA: ACM Press, 2016: 2073-2083.

[13] ALLAMANIS M, PENG H, SUTTON C. A convolutional attention network for extreme summarization of source code[C]//Proceedings of International Conference on Machine Learning. New York, USA: ACM Press, 2016: 2091-2100.

[14] WAN Yao, ZHAO Zhou, YANG Min, et al. Improving automatic source code summarization via deep reinforcement learning[C]//Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering. New York, USA: ACM Press, 2018: 397-407.

[15] HU Xing, LI Ge, XIA Xin, et al. Deep code comment generation[C]//Proceedings of the 26th Conference on Program Comprehension. New York, USA: ACM Press, 2018: 200-210.

[16] XU Kun, WU Lingfei, WANG Zhiguo, et al. Graph2Seq: graph to sequence learning with attention-based neural networks[EB/OL]. [2019-12-20]. https://arxiv.org/pdf/1804.00823.pdf.

[17] SEE A, LIU P J, MANNING C D. Get to the point: summarization with pointer-generator networks [C]// Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics. New York, USA: ACM Press, 2017: 1073-1083.

[18] GU Jiatao, LU Zhengdong, LI Hang, et al. Incorporating copying mechanism in sequence-to-sequence learning[C]// Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics. New York, USA: ACM Press, 2016: 1631-1640.

[19] WANG L, BLUNSOM P, GREFENSTETTE E, et al. Latent predictor networks for code generation[C]//Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics. New York, USA: ACM Press, 2016: 599-609.

[20] LIANG Y D, ZHU K Q. Automatic generation of text descriptive comments for code blocks[EB/OL]. [2019-12-20]. http://www. cs. sjtu. edu. cn/~kzhu/papers/kzhu-aaai 18-code. pdf.

[21] LUONG M T, PHAM H, MANNING C D. Effective approaches to attention-based neural machine translation[C]// Proceedings of 2015 Conference on Empirical Methods in Natural Language Processing. Washington D. C. , USA: IEEE Press, 2015: 1412-1421.

[22] ZHONG V, XIONG C M, SOCHER R. Seq2SQL: generating structured queries from natural language using reinforcement learning[EB/OL]. [2019-12-20]. https://arxiv. org/pdf/1709. 00103. pdf.

[23] DONG L, LAPATA M. Language to logical form with neural attention [C]//Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics. New York, USA: ACM Press, 2016: 33-47.

[24] YIN P C, NEUBIG G. A syntactic neural model for general-purpose code generation [C]//Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics. New York, USA: ACM Press, 2017: 440-458.

[25] RABINOVICH M, STERN M, KLEIN D. Abstract syntax networks for code generation and semantic parsing[C]// Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics. New York, USA: ACM Press, 2017: 1139-1153.

[26] ERIGUCHI A, HASHIMOTO K, TSURUOKA Y. Tree-to-sequence attentional neural machine translation [C]// Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics. New York, USA: ACM Press, 2016: 823-846.

[27] PAPINENI K, ROUKOS S, WARD T, et al. BLEU: a method for automatic evaluation of machine translation[C]// Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics. New York, USA: ACM Press, 2002: 311-318.

[28] LIN C Y. Rouge: a package for automatic evaluation of summaries[EB/OL]. [2019-12-20]. https://www. aclweb. org/anthology/W04-1013. pdf.

[29] KINGMA D P, BA J. Adam: a method for stochastic optimization[EB/OL]. [2019-12-20]. https://arxiv. org/pdf/1412. 6980. pdf.

[30] MIKOLOV T, CHEN K, CORRADO G, et al. Efficient estimation of word representations in vector space[EB/OL]. [2019-12-20]. https://arxiv. org/pdf/1301. 3781. pdf.

[31] PENNINGTON J, SOCHER R, MANNING C. Glove: global vectors for word representation[EB/OL]. [2019-12-20]. https://nlp. stanford. edu/pubs/glove. pdf.

[32] GLOROT X, BENGIO Y. Understanding the difficulty of training deep feedforward neural networks [C]// Proceedings of the 13th International Conference on Artificial Intelligence and Statistics. New York, USA: ACM Press, 2010: 249-256.

编辑　吴云芳