·人工智能与模式识别·　　文章编号：1000-3428(2023)02-0061-09　　文献标志码：A　　中图分类号：TP312

基于结构感知混合编码模型的代码注释生成方法

蔡瑞初，张盛强，许柏炎

（广东工业大学 计算机学院，广州 510006）

摘　要：代码注释能够提高程序代码的可读性，从而提升软件开发效率并降低成本。现有的代码注释生成方法将程序代码的序列表示或者抽象语法树表示输入到不同结构的编码器网络，无法融合程序代码不同抽象形式的结构特性，导致生成的注释可读性较差。构建一种结构感知的混合编码模型，同时考虑程序代码的序列表示和结构表示，通过序列编码层和图编码层分别捕获程序代码的序列信息和语法结构信息，并利用聚合编码过程将两类信息融合至解码器。设计一种结构感知的图注意力网络，通过将程序代码的语法结构的层次和类型信息嵌入图注意力网络的学习参数，有效提升了混合编码模型对程序代码的复杂语法结构的学习能力。实验结果表明，与SiT基准模型相比，混合编码模型在Python和Java数据集上的BLEU、ROUGE-L、METEOR得分分别提高了2.68%、1.47%、3.82%和2.51%、2.24%、3.55%，能生成更准确的代码注释。

关键词：代码注释生成；混合编码模型；图注意力网络；深度自注意力网络；自然语言处理

开放科学（资源服务）标志码（OSID）：

中文引用格式：蔡瑞初，张盛强，许柏炎.基于结构感知混合编码模型的代码注释生成方法[J].计算机工程，2023，49(2)：61-69.

英文引用格式：CAI R C，ZHANG S Q，XU B Y.Method for generating code comments based on structure-aware hybrid encoding model[J].Computer Engineering，2023，49(2)：61-69.

Method for Generating Code Comments Based on Structure-aware Hybrid Encoding Model

CAI Ruichu，ZHANG Shengqiang，XU Boyan

（School of Computers，Guangdong University of Technology，Guangzhou 510006，China）

【Abstract】Code comments improve the readability of program codes，enhancing software development efficiency and reducing costs.Existing methods for code comment generation feed the sequence form or Abstract Syntax Tree（AST）form of a program code into encoder networks with different structures，which cannot fuse the structural characteristics of different abstract forms of program codes.This results in poor readability of the generated comments.This study proposes a Structure-aware Hybrid Encoding（SHE）model.The SHE model considers both the sequence form and structure form of the program code.This includes capturing the context information and the grammar structure information of the program code by the sequence encoding layer and the graph encoding layer，respectively，and effectively fusing the above two aspect information into the decoder through aggregation encoding.This study further proposes a Structure-aware Graph Attention（SGAT）network to effectively improve the learning ability of the SHE model for the complex grammar structure of a program code by integrating the hierarchical and type information of the grammar structure of the program code into the learning parameters of the graph attention network.The experimental results show that compared with the Structure-induced Transformer（SiT）baseline models，the SHE model improves the Bi-Lingual Evaluation Understudy（BLEU），Recall-Oriented Understudy for Gisting Evaluation-Longest common subsequence（ROUGE-L），and Metric for Evaluation of Translation with Explicit Ordering（METEOR）scores by 2.68%，1.47%，and 3.82%，respectively，on the Python dataset.Moreover，the SHE model improves BLEU，ROUGE-L and METEOR scores by 2.51%，2.24%，and 3.55%，respectively，on the Java dataset.The experimental results demonstrate that the SHE model can generate more accurate code comments than the baseline models.

【Key words】code comment generation；hybrid encoding model；graph attention network；deep self-attention network；natural language processing

DOI：10. 19678/j. issn. 1000-3428. 0063592

基金项目：国家自然科学基金（61876043）；国家优秀青年科学基金（62122022）；广州市科技计划项目（201902010058）。

作者简介：蔡瑞初（1983—），男，教授、博士、博士生导师，主研方向为因果关系、机器学习；张盛强，硕士研究生；许柏炎（通信作者），博士研究生。

收稿日期：2021-12-21　　**修回日期：**2022-02-18　　**E-mail：**hpakyim@gmail.com

0 概述

代码注释是程序源代码功能含义的可读注解，在软件维护过程中起到关键作用[1-2]，然而由于编写代码注释的时间成本较高，软件项目中经常出现注释不全、不清晰，甚至无注释等情况。为了解决上述问题，软件工程社区提出了代码注释自动生成任务，其目标是通过机器学习的方法自动将程序代码转化为自然语言注释。代码注释生成作为自然语言生成[3-5]的一项重要任务，在近年来受到广泛关注[6-7]。

现有代码注释生成方法一般将源代码表示为序列形式或者抽象语法树（Abstract Syntax Tree，AST）形式。早期经典的研究方法[8]使用循环神经网络作为编码器对源代码序列进行建模。这种方法在建模过程中只关注源代码的序列信息，忽略了源代码的结构信息，导致生成的注释质量较差。为了捕获源代码中的结构信息，文献[9]将源代码表示为AST中的一系列组合路径，并对每条路径进行编码。文献[10]提出一种基于结构的遍历方式来遍历AST，以获得更多的结构信息。文献[11]将源代码转化为多视角图，并将结构信息融入深度自注意力网络（Transformer）[12]。以上方法虽然从不同结构形式对源代码的结构特征进行建模，但是缺少对不同结构形式的融合过程，也缺乏充分考虑程序代码的语法结构的类型和层次信息，使得在面对复杂程序代码时生成的注释存在准确率低和可读性差的问题。

针对上述结构编码方面的缺陷，本文建立一种结构感知的混合编码（Structure-aware Hybrid Encoding，SHE）模型，包括序列编码、语法结构编码和聚合编码三层编码过程，能够同时捕获源代码的节点序列信息和语法结构信息。设计一种结构化感知的图注意力（Structure-aware Graph Attention，SGAT）网络作为SHE的语法结构编码层，考虑了源代码的语法结构的层次和类型信息，以提升模型对复杂代码的结构学习能力。

1 相关工作

根据对源代码的处理方式，可将现有的研究方法分为三类：基于序列的方法，基于树的方法和基于图的方法。

基于序列的方法直接将源代码转化为序列结构形式。文献[8]将源代码转化为序列，并使用带有注意力机制[13]的循环神经网络来生成注释。文献[14]使用两个编码器分别学习源代码的序列信息和API的序列信息，并在解码时通过API知识辅助注释的生成。

基于树的方法通过语法解析器将源代码转化为AST。为了捕获源代码中的结构信息，文献[9]将源代码表示为AST中的一系列组合路径，并在解码时使用注意力机制来选择相关路径。文献[15]所提出的模型采用两个门控循环单元（Gated Recurrent Unit，GRU）[16]分别对源代码序列和AST进行编码，并在解码时通过融合源代码的序列信息和AST的结构信息来构建上下文向量。文献[17]提出一种用于代码注释生成的编码器-解码器框架，该框架将源代码视为N叉树并通过与类型相关的编码器和受类型限制的解码器以捕获更多的结构信息。

基于图的方法将源代码转化为图的形式。文献[18]将门控图神经网络（Gated Graph Neural Network，GGNN）[19]融入现有的序列编码器，使得模型能够捕获弱结构化数据中的长距离依赖关系。文献[20]使用图神经网络（Graph Neural Network，GNN）[21]来提取源代码的结构特征。文献[22]将源代码转化为基于注意力机制的动态图，并设计一种混合型GNN来捕获局部和全局结构信息。

与上述三种结构编码研究方法不同，本文将源代码转化为程序图，并提出一种基于结构感知的混合编码模型（SHE）。通过融合序列编码层和图编码层以更好地捕获源代码的节点序列信息和语法结构信息。

在代码修复任务上，文献[23-24]均采用了序列模型和图模型堆叠的三层编码过程。但是，本文与上述工作采用了不同的序列编码层或图网络编码层。更进一步地，代码注释生成对比代码修复需要更好地理解代码的语法结构中蕴含的语义信息。本文提出的结构化感知的图注意力网络（SGAT）通过更好地考虑源代码的语法结构的层次和类型信息，以对复杂代码有更好的学习能力。

2 混合编码模型构建

2.1 问题定义

首先对代码注释生成任务进行必要的公式化定义。当给定一段源代码C时，代码注释生成任务的目标是生成能够准确描述该源代码的自然语言注释$Y=\{y_1,y_2,\cdots,y_{|M|}\}$。为了更好地捕获源代码的上下文信息和结构信息，同时考虑了源代码的序列表示和抽象语法树（AST）表示。长度为N的源代码序列表示为$C=\{w_1,w_2,\cdots,w_{|N|}\}$，其中$w_i$代表源代码中的第$i$个词。源代码$C$的AST表示需要通过语法解析器得到。将AST树形表示进一步构建成程序图表示G。

2.2 程序图构建

程序图$G=\{V,E\}$。在程序图G上，V表示节点v_i的集合，节点$v_i=\{w_i,\tau_i\}$具有节点值w_i和节点类型τ_i。节点构建过程如下：首先，通过语法解析器将源代码转化为AST树形表示，AST包含非终端节点和终端节点，其中，非终端节点代表特定的语法规则，终端

节点代表文本标记（token），如标识符、字符串和数字；然后，所有AST上的节点都分配有对应的节点值和节点类型，即w_i和τ_i。值得注意的是，当终端节点的节点值由多个单词构成时，将该终端节点拆分为多个类型相同的子节点。在程序图G上，E表示边$\{v_i, v_j, l_{ij}\}$的集合，l_{ij}表示节点v_i和v_j所连接的边类型。不同类型的边构建过程如下：首先，将default语法边表示为AST中节点的连接关系，用于语法信息的学习；然后，在AST中引入多种具有语义依赖信息的边。使用NextNode边连接兄弟节点，以便信息能够在兄弟节点之间以更近的距离传播。使用SameToken边连接所有具有相同节点值的节点，使得信息能够跨长距离传播。此外，为每条边添加相应的反向边（reverse），并为每个节点添加相应的自循环边（self-loop），以便对反向依赖和自依赖进行建模。

节点类型和语法边是由对应编程语言的语法解析器自动生成的，而具有语义依赖信息的边的构建过程是通用的。因此，本文提出的程序图构建过程具有通用性，可快速扩展到不同的编程语言。以Python源代码为例，其对应的程序图和注释如图1所示。

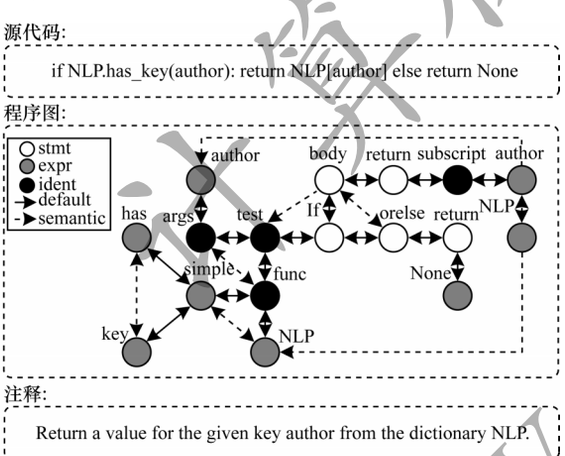

图1　Python源代码对应的程序图和注释

Fig.1　Corresponding program graph and comment for Python source code

2.3　模型架构

本文提出的SHE模型整体架构如图2所示，该模型包含以下三部分：程序图向量化，结构感知混合编码器和解码器。

在程序图G向量化阶段，主要对程序图的节点进行初始向量化操作，并作为结构感知混合编码器的输入。

在结构感知混合编码器阶段，主要包括了序列编码、语法结构编码和聚合编码三层编码过程。其核心是通过融合序列编码层和图编码层以更好地捕获源代码的上下文信息和语法结构信息。同时，为了更好地学习语法结构信息，本文设计的结构化感知的图注意力（SGAT）网络将节点类型和边类型的信息引入网络结构。

在解码器阶段，主要应用Transformer和复制机制[25]有效避免自然语言生成任务中经典的未登录词（Out of Vocabulary，OOV）问题。

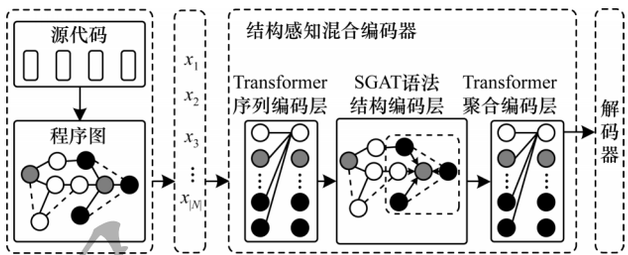

图2　SHE模型的整体架构

Fig.2　Overall architecture of SHE model

2.4　程序图向量化

首先，创建词典的词嵌入矩阵$\boldsymbol{D}$，该词典可通过词id进行查找。其次，将节点值映射到词典中的词id。根据词id可得到节点值嵌入$\boldsymbol{x}_i^{\mathrm{VE}}$：

$$\boldsymbol{x}_i^{\mathrm{VE}} = \mathrm{Lookup}(\boldsymbol{D}, \mathrm{Map}(w_i)) \tag{1}$$

其中：Map表示将词w_i映射为词id的函数；Lookup表示词嵌入查找函数。

SHE的Transformer序列编码层通过注意力机制学习节点序列的表示，但无法自动捕获节点序列的位置信息。因此，需要将位置信息融入输入特征。根据节点v_i所在的位置i可得到节点位置嵌入$\boldsymbol{x}_i^{\mathrm{PE}}$。

$$P_{(i,k)}^{\mathrm{PE}} = \sin\left(\frac{i}{10\,000^{k/d_p}}\right), k = 2j \tag{2}$$

$$P_{(i,k)}^{\mathrm{PE}} = \cos\left(\frac{i}{10\,000^{(k-1)/d_p}}\right), k = 2j+1 \tag{3}$$

$$\boldsymbol{x}_i^{\mathrm{PE}} = \left[P_{(i,0)}^{\mathrm{PE}}, P_{(i,1)}^{\mathrm{PE}}, \cdots, P_{(i,d_p-1)}^{\mathrm{PE}}\right] \tag{4}$$

其中：d_p表示位置嵌入的维度；$P_{(i,k)}^{\mathrm{PE}}$表示节点v_i的位置嵌入在第k维的值，当k为偶数时使用式（2）计算$P_{(i,k)}^{\mathrm{PE}}$，当k为奇数时使用式（3）计算$P_{(i,k)}^{\mathrm{PE}}$。

将节点值嵌入和位置嵌入相加后，可得到节点的输入特征$\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_{|N|}\}$。

$$\boldsymbol{x}_i = \boldsymbol{x}_i^{\mathrm{VE}} + \boldsymbol{x}_i^{\mathrm{PE}} \tag{5}$$

2.5　结构感知混合编码器

2.5.1　Transformer序列编码层

源代码片段长度一般较长，经典的长短期记忆网络无法很好地捕获源代码序列中的长期依赖关系。因此，采用Transformer来提取源代码的上下文信息。Transformer序列编码层由多个相同的层堆叠而成，并在每一层中应用多头注意力机制（multi-head attention）。将节点输入特征$\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_{|N|}\}$输送到Transformer序列编码层，可得到节点的输出特征$\boldsymbol{P} = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \cdots, \boldsymbol{p}_{|N|}\}$。计算过程如式（6）~式（10）所示：

$$\boldsymbol{e}_{ij}^{(h)} = \frac{\left(\boldsymbol{x}_i \boldsymbol{W}_Q^{(h)}\right)\left(\boldsymbol{x}_j \boldsymbol{W}_K^{(h)}\right)^{\mathrm{T}}}{\sqrt{d_k}} \tag{6}$$

$$\boldsymbol{\alpha}_{ij}^{(h)}=\frac{\exp(\boldsymbol{e}_{ij}^{(h)})}{\sum_{w=1}^{N}\exp(\boldsymbol{e}_{iw}^{(h)})} \tag{7}$$

$$\boldsymbol{z}_i^{(h)}=\sum_{j=1}^{N}\boldsymbol{\alpha}_{ij}^{(h)}\left(\boldsymbol{x}_j\boldsymbol{W}_V^{(h)}\right) \tag{8}$$

$$\boldsymbol{z}_i=\mathrm{LayerNorm}\left(\boldsymbol{x}_i+\left[\boldsymbol{z}_i^{(1)},\boldsymbol{z}_i^{(2)},\cdots,\boldsymbol{z}_i^{(H)}\right]\right) \tag{9}$$

$$\boldsymbol{p}_i=\mathrm{LayerNorm}\left(\boldsymbol{z}_i+\mathrm{FFN}(\boldsymbol{z}_i)\right) \tag{10}$$

其中：$\boldsymbol{W}_Q^{(h)}$、$\boldsymbol{W}_K^{(h)}$和$\boldsymbol{W}_V^{(h)}$表示第h个注意力机制对应的权重矩阵；d_k表示每个头的维度；$\boldsymbol{e}_{ij}^{(h)}$和$\boldsymbol{\alpha}_{ij}^{(h)}$表示由第$h$个注意力机制计算得到的相关系数和归一化注意力系数；$\boldsymbol{z}_i^{(h)}$表示由第h个注意力机制计算得到的输出；H表示多头注意力机制的头数；LayerNorm表示层归一化操作；[ ]表示拼接操作；FFN表示前馈神经网络。

Transformer的计算公式可简化如下：

$$\boldsymbol{P}=\mathrm{Transformer}(\boldsymbol{X}) \tag{11}$$

2.5.2 SGAT语法结构编码层

SHE的语法结构编码层负责捕获源代码的复杂结构信息。现有的工作一般只考虑抽象语法树(AST)中节点的连接关系和层次信息，而忽略编码中的语法类型信息，导致无法很好地区分结构相似但语义不同的子树或者子图。源代码的语法类型包含不同的语义信息。通过引入语法类型信息，使得模型能够区分上述复杂语法结构。由于现有的图注意力网络无法很好地适用于多节点类型和多边类型的情况，因此本文提出结构化感知的图注意力网络(SGAT)用于程序图的结构学习。

将节点类型信息和边类型信息作为SGAT语法结构编码层的可学习参数，而不是直接将节点类型信息和边类型信息作为特征输入编码器，使得SGAT能够更好地学习不同的语法结构表示，提升模型对复杂程序的结构学习能力。具体地，对于程序图G中的节点v_i，分别计算其与邻居节点v_j的相关系数$\boldsymbol{o}_{ij}$：

$$\boldsymbol{o}_{ij}=a\left(\left[\boldsymbol{W}_{\tau_i}\boldsymbol{p}_i,(\boldsymbol{W}_{\tau_j}\boldsymbol{p}_j+\boldsymbol{r}_{l_{ij}})\right]\right),j\in\mathcal{N}_i \tag{12}$$

其中：$\boldsymbol{p}_i$和$\boldsymbol{p}_j$表示经过Transformer序列编码层优化后的节点特征；τ_i和τ_j表示节点的类型；l_{ij}表示节点之间的边类型；$\mathcal{N}_i$表示节点v_i的邻居节点集合；$\boldsymbol{W}_{\tau_i}$和$\boldsymbol{W}_{\tau_j}$表示与节点类型相关的权重矩阵；$\boldsymbol{r}_{l_{ij}}$表示与边类型对应的向量；a表示注意力机制，用于计算相关系数。

进一步地，将相关系数$\boldsymbol{o}_{ij}$进行归一化，可得到归一化注意力系数$\boldsymbol{g}_{ij}$：

$$\boldsymbol{g}_{ij}=\frac{\exp\left(\mathrm{LeakyReLU}\left(\boldsymbol{o}_{ij}\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\mathrm{LeakyReLU}\left(\boldsymbol{o}_{ik}\right)\right)} \tag{13}$$

其中：LeakyReLU表示非线性激活函数；exp表示以自然常数e为底的指数函数。

在得到归一化注意力系数$\boldsymbol{g}_{ij}$后，对特征进行加权求和操作，可得到节点的输出特征$\boldsymbol{Q}=\{\boldsymbol{q}_1,\boldsymbol{q}_2,\cdots,\boldsymbol{q}_{|N|}\}$。

$$\boldsymbol{q}_i=\sigma\left(\sum_{j\in\mathcal{N}_i}\boldsymbol{g}_{ij}\left(\boldsymbol{W}_{\tau_j}\boldsymbol{p}_j+\boldsymbol{r}_{l_{ij}}\right)\right) \tag{14}$$

其中：$\boldsymbol{W}_{\tau_j}$和$\boldsymbol{r}_{l_{ij}}$表示注意力机制所对应的权重矩阵和向量；σ表示非线性激活函数。

同时，采用多头注意力机制进一步提升模型的表现能力，具体如下：

$$\boldsymbol{q}_i=\Big\|_{h=1}^{H}\sigma\left(\sum_{j\in\mathcal{N}_i}\boldsymbol{g}_{ij}^{(h)}\left(\boldsymbol{W}_{\tau_j}^{(h)}\boldsymbol{p}_j+\boldsymbol{r}_{l_{ij}}^{(h)}\right)\right) \tag{15}$$

其中：H表示多头注意力机制的头数；$\|$表示拼接操作。

SGAT的计算公式可简化如下：

$$\boldsymbol{Q}=\mathrm{SGAT}(\boldsymbol{P}) \tag{16}$$

SGAT通过对程序图中不同的节点类型和边类型引入可学习的参数$\boldsymbol{W}_{\tau_j}^{(h)}$和$\boldsymbol{r}_{l_{ij}}^{(h)}$，使得模型能够区分不同的节点类型和边类型的语义信息，因此能够更准确地捕获源代码中的结构信息。同时，通过多头注意力机制使得学习过程更加稳定，进一步提升了模型性能。

2.5.3 Transformer聚合编码层

Transformer序列编码层通过多头注意力机制来感知节点序列的全局信息，SGAT语法结构编码层基于图的信息传播来捕获程序图的局部结构信息。为了聚合代码的上下文信息和语法结构信息，SHE采用Transformer聚合编码层有效地聚合前两层编码的结构表示并输出到解码器。通过该聚合编码层，可获得节点的最终输出特征$\boldsymbol{U}=\{\boldsymbol{u}_1,\boldsymbol{u}_2,\cdots,\boldsymbol{u}_{|N|}\}$。

$$\boldsymbol{Q}=\boldsymbol{P}+\mathrm{SGAT}(\boldsymbol{P}) \tag{17}$$

$$\boldsymbol{U}=\mathrm{Transformer}(\boldsymbol{Q}) \tag{18}$$

首先，将节点序列的初始输入特征输送到Transformer序列编码层，以提取源代码中的序列信息。其次，将优化后的特征输送到SGAT语法结构编码层，以捕获源代码的复杂结构信息。为了防止网络退化问题，式(17)使用了残差连接。最后，采用Transformer聚合编码层聚合前两层的信息并输出到解码器。

2.6 解码器

解码阶段采用Transformer解码器来生成代码注释。同时，引入复制机制以提高生成的注释的质量。该解码器包含两种注意力机制：一种是基于掩码的多头注意力机制；另一种是基于编码器-解码器的多头注意力机制。基于掩码的多头注意力机制仅用于解码器，其允许解码器结合之前输出的单词序列来决定当前阶段所输出的单词。在训练过程中，为了防止未来信息的泄露，需要遮掩未来阶段的单词。基于编码器-解码器的多头注意力机制则用于编码

器和解码器之间，其能够让解码器捕捉到编码器的输出信息。

通过Transformer解码器，可得到解码阶段t的目标词典的概率分布P_{v}：

$$P_{\mathrm{v}}=\mathrm{Softmax}\left(\boldsymbol{W}_{\mathrm{gen}}\boldsymbol{s}_t\right) \tag{19}$$

其中：Softmax表示归一化指数函数；$\boldsymbol{W}_{\mathrm{gen}}$表示可学习的参数矩阵；$\boldsymbol{s}_t$表示解码阶段输出的隐藏状态。

在构建目标词典时，通常会过滤掉出现频次较低的单词，以限制目标词典的大小。这些出现频次较低的单词不存在于目标词典中，被称为未登录词（OOV）。但是，源代码中表示数值或字符的低频词包含重要的信息，其经常出现在目标注释中。如果忽略了包含重要信息的低频词，将使得生成的注释出现准确率降低的问题。为了缓解OOV现象，采用复制机制以一定的概率从输入中复制单词。

在解码阶段t，计算隐藏状态$\boldsymbol{s}_t$与聚合编码层的输出特征$\boldsymbol{U}=\{\boldsymbol{u}_1,\boldsymbol{u}_2,\cdots,\boldsymbol{u}_{|N|}\}$的归一化注意力系数$\boldsymbol{f}_{ti}$：

$$\boldsymbol{f}_{ti}=\mathrm{Softmax}\left(\mathrm{score}\left(\boldsymbol{u}_i,\boldsymbol{s}_t\right)\right) \tag{20}$$

其中：score表示计算相关系数的函数。

在得到归一化注意力系数$\boldsymbol{f}_{ti}$后，计算生成概率$p_{\mathrm{g}}\in[0,1]$：

$$\boldsymbol{c}_t=\sum_{i=1}^{N}\boldsymbol{f}_{ti}\boldsymbol{u}_i \tag{21}$$

$$p_{\mathrm{g}}=\mathrm{Sigmoid}\left(\boldsymbol{W}_{\mathrm{ptr}}[\boldsymbol{c}_t,\boldsymbol{s}_t,\boldsymbol{d}_t]\right) \tag{22}$$

其中：$\boldsymbol{W}_{\mathrm{ptr}}$表示可学习的参数矩阵；$\boldsymbol{c}_t$表示通过加权求和得到的上下文向量；$\boldsymbol{d}_t$表示当前解码阶段的输入；Sigmoid表示非线性激活函数。

由生成概率p_{g}可得到解码阶段t的目标词典的概率分布$P_{\mathrm{fin}}(m)$：

$$P_{\mathrm{fin}}(m)=p_{\mathrm{g}}P_{\mathrm{v}}(m)+\left(1-p_{\mathrm{g}}\right)\sum_{i;\,w_i=m}\boldsymbol{f}_{ti} \tag{23}$$

其中：m表示目标词典中的某个单词；w_i表示节点v_i的节点值。因此，每个单词的最终输出概率由其生成概率和复制概率共同决定。

通过最小化交叉熵损失函数对模型进行训练：

$$L_{\mathrm{Loss}}=\frac{1}{T}\sum_{t=1}^{T}-\ln P_{\mathrm{fin}}(y_t) \tag{24}$$

其中：y_t表示解码阶段t的目标单词。

3　实验结果与分析

3.1　数据集

实验数据采用文献[26]提供的Python数据集和文献[14]提供的Java数据集。Python数据集包含55 538个训练样本、18 505个验证样本和18 502个测试样本。Java数据集包含69 708个训练样本、8 714个验证样本和8 714个测试样本。使用相应的语法解析器将源代码转化为AST。对于Python源代码，生成的AST包含14种节点类型。对于Java源代码，生成的AST包含15种节点类型。在AST的基础上，通过添加多种具有语义依赖信息的边来构建程序图。

3.2　评价指标

为了更准确地评估模型效果，实验采用BLEU[27]、ROUGE[28]和METEOR[29]三种被广泛使用的评价指标从不同角度考虑生成的注释的质量。BLEU与ROUGE类似，但BLEU侧重于计算准确率，而ROUGE更加关注召回率。METEOR通过引入同义词集，在计算得分时考虑了同义词及词性变换的情况。

3.3　参数设置

实验基于PyTorch框架在RTX 3090 GPU上进行训练。在训练过程中，使用Adam[30]作为优化器，并使用0.000 05作为初始学习率。在权重初始化过程中，使用Xavier[31]作为初始化器。对于编码器和解码器，多头注意力机制的隐藏层单元数设置为64，头数设置为8，单词嵌入的维度设置为512。在编码过程中，节点的最大数量设置为400。在解码过程中，生成的注释的最大长度设置为50。

3.4　基准模型

为了全面评估SHE的性能，选用多个具有代表性的模型进行对比和分析。

1）CODE-NN[8]是第一个使用端到端的方式将源代码转化为注释的模型，通过使用带有注意力机制的LSTM[32]来解决代码注释生成任务，并取得了显著的效果。

2）DeepCom[10]通过基于结构的遍历方式获得模型所需的输入序列。相比于通过前序遍历方式所获得的序列，基于结构的遍历方式所获得的序列可还原为原始的AST，在提取结构信息上更加有效。

3）Tree2Seq[33]使用基于树的LSTM作为编码器，以显式地捕获结构信息。

4）RL+Hybrid2Seq[26]利用LSTM和基于AST的LSTM来分别提取源代码的序列特征和结构特征，并采用强化学习的方式来训练模型。

5）API+CODE[14]通过源代码中的API知识来辅助代码注释生成。

6）Dual Model[34]将代码生成和代码注释生成作为对偶任务，进一步提升了生成的注释的质量。

7）Trans[35]是一种用于表示源代码中的标记之间的相对位置的方法，并在解码过程中应用了复制机制来辅助注释生成。

8）SiT[11]是一种Transformer的变体。SiT将源代码转化为多视角图，并将结构信息融入Transformer。

3.5　结果分析

3.5.1　模型性能分析

不同模型在Python数据集和Java数据集上的实验结果如表1和表2所示，其中加粗表示最优的结果。CODE-NN的评价指标得分低于其他模型，因为

其没有建立语言模型，不能很好地捕获源代码的序列信息和结构信息。DeepCom、Tree2Seq和RL+Hybrid2Seq通过不同方式对源代码进行建模，使得模型能够捕获源代码中的结构信息。API+CODE通过API知识指导注释的生成，在Java数据集上取得了较高的得分。Dual Model将代码生成任务和代码注释生成任务视为对偶任务，进一步提升了生成的注释的质量。Trans和SiT使用Transformer来学习源代码的表示，提升了模型的表现能力。

表1 不同模型在Python数据集上的性能对比结果

Table 1 Performance comparison results of different models on Python dataset %

模型	BLEU	ROUGE-L	METEOR
CODE-NN	17.36	37.81	9.29
DeepCom	20.78	37.35	9.98
Tree2Seq	20.07	35.64	8.96
RL+Hybrid2Seq	19.28	39.34	9.75
API+CODE	15.36	33.65	8.57
Dual Model	21.80	39.45	11.14
Trans	32.85	46.93	19.86
SiT	33.84	48.14	20.67
SHE	**34.75**	**48.85**	**21.46**

表2 不同模型在Java数据集上的性能对比结果

Table 2 Performance comparison results of different models on Java dataset %

模型	BLEU	ROUGE-L	METEOR
CODE-NN	27.60	41.10	12.61
DeepCom	39.75	52.67	23.06
Tree2Seq	37.88	51.50	22.55
RL+Hybrid2Seq	38.22	51.91	22.75
API+CODE	41.31	52.25	23.73
Dual Model	42.39	53.61	25.77
Trans	44.87	54.95	26.58
SiT	45.76	55.58	27.58
SHE	**46.91**	**56.83**	**28.56**

由表1和表2可知，SHE在Python数据集和Java数据集上的评价指标得分均高于所有基准模型。与SiT模型相比，SHE在Python数据集上的BLEU、ROUGE-L和METEOR得分分别提高了2.68%、1.47%和3.82%，在Java数据集上的BLEU、ROUGE-L和METEOR得分分别提高了2.51%、2.24%和3.55%。实验结果表明，SHE能够更好地捕获源代码的节点序列信息和复杂的语法结构信息。

3.5.2 模型收敛速度分析

模型的收敛速度决定模型达到最佳性能时所需时间的长短。不同模型在Java训练集上的收敛速度对比结果如图3所示。通过设置ROUGE-L和BLEU性能参考线观察，SHE需要约90个训练轮数到达参考线，而SiT则需要约175个训练轮数。可见，SHE具有结构化感知的混合编码结构，对复杂语法结构有更高效的学习能力，模型有更快的收敛速度。

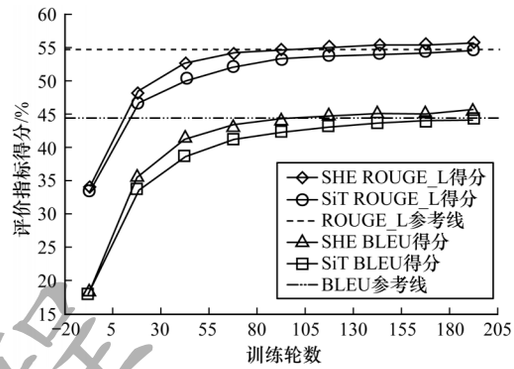

图3 不同模型在Java训练集上的收敛速度对比结果

Fig.3 Comparison results of convergence speed of different models on Java training set

3.5.3 模型参数分析

多头注意力机制的头数和隐藏层单元数是影响模型性能的重要参数。为方便对比，将所有编码层的多头注意力机制的头数和隐藏层单元数均设置为相同的值。

不同头数的模型在Python数据集上的实验结果如表3所示，当多头注意力机制的隐藏层单元数设置为64时，随着头数的增加，模型的性能越来越好，这说明在一定程度上增加头数能够使得网络从不同子空间捕获到更丰富的特征。

表3 不同头数的模型在Python数据集上的实验结果

Table 3 Experimental results of models with different number of heads on Python dataset %

头数	BLEU	ROUGE-L	METEOR
1	32.96	46.68	19.79
2	33.55	47.50	20.37
4	33.82	47.89	20.53
8	34.75	48.85	21.46

如表4所示，在多头注意力机制的头数设置为8的情况下，随着隐藏层单元数的增加，模型在不同评价指标上的得分也越来越高，这说明在一定程度上增加隐藏层单元数能够提升网络的表现能力。

表4 不同隐藏层单元数的模型在Python数据集上的实验结果

Table 4 Experimental results of models with different number of hidden layer units on Python dataset %

隐藏层单元数	BLEU	ROUGE-L	METEOR
8	33.22	47.07	20.02
16	33.62	47.76	20.43
32	33.71	48.02	20.65
64	34.75	48.85	21.46

3.5.4　代码长度分析

代码长度是衡量模型表现能力的重要因素之一。不同代码长度的模型在Python数据集上的实验结果如图4所示。

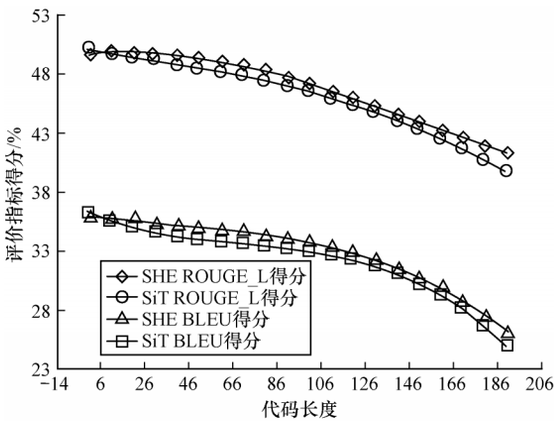

图4　不同代码长度的模型在Python数据集上的实验结果

Fig.4　Experimental results of models with different code lengths on Python dataset

随着代码长度的增加，不同模型所取得的BLEU得分和ROUGE_L得分均呈下降趋势，这说明代码长度越长，模型越难捕获源代码的完整信息。在相同代码长度的情况下，SHE所取得的BLEU得分和ROUGE_L得分高于SiT，这说明SHE对于更复杂的程序代码仍然具有良好的结构学习能力。

3.5.5　节点排列顺序分析

程序图中节点的排列顺序与AST中节点的排列顺序一致。通过不同的遍历方式可得到不同的节点排列顺序。为了探究程序图中节点的排列顺序对模型性能的影响，设计相关对比实验。不同的节点排列顺序在Python数据集和Java数据集上的实验结果如表5和表6所示。由表5和表6可知，在Python数据集和Java数据集上，对于SiT和SHE，按不同遍历方式排列所取得的指标得分在相应的模型中均没有明显的差距，这可能是因为按不同遍历方式所得到的节点序列的排列顺序虽然不同，但仍然包含AST中的有效信息。

表5　不同节点排列顺序的模型在Python数据集上的实验结果

Table 5　Experimental results of models with different node arrangement orders on Python dataset　%

模型	排列方式	BLEU	ROUGE-L	METEOR
SiT	按前序遍历排列	33.84	48.14	20.67
	按后序遍历排列	33.81	48.09	20.65
	按层序遍历排列	33.79	48.07	20.64
SHE	按前序遍历排列	34.75	48.85	21.46
	按后序遍历排列	34.72	48.83	21.46
	按层序遍历排列	34.73	48.82	21.45

表6　不同节点排列顺序的模型在Java数据集上的实验结果

Table 6　Experimental results of models with different node arrangement orders on Java dataset　%

模型	排列方式	BLEU	ROUGE-L	METEOR
SiT	按前序遍历排列	45.76	55.58	27.58
	按后序遍历排列	45.73	55.54	27.56
	按层序遍历排列	45.69	55.51	27.55
SHE	按前序遍历排列	46.91	56.83	28.56
	按后序遍历排列	46.87	56.80	28.54
	按层序遍历排列	46.85	56.78	28.54

3.6　消融实验分析

为了进一步探究模型各组成部分的作用，设计消融实验。在Python数据集上的消融实验结果如表7所示，其中，加粗表示最优的结果，No-SGAT表示仅考虑源代码的节点序列信息，No-ET-NT表示不区分程序图的节点类型和边类型，No-ET表示不考虑程序图的边类型信息，No-NT表示不考虑程序图的节点类型信息，No-AEL表示不采用Transformer聚合编码层。

表7　在Python数据集上的消融实验结果

Table 7　Results of ablation experiments on Python dataset　%

模型	BLEU	ROUGE-L	METEOR
No-SGAT	33.82	48.14	20.66
No-ET-NT	34.12	48.28	20.86
No-ET	34.56	48.60	21.26
No-NT	34.50	48.56	21.23
No-AEL	34.48	48.40	21.25
SHE	**34.75**	**48.85**	**21.46**

由表7可以看出：仅考虑源代码的节点序列信息，将导致模型的性能下降；在不区分程序图的节点类型和边类型的情况下，模型取得的评价指标得分均有所提升，这说明将源代码转化为程序图能够提高模型的表现能力；在考虑节点类型信息或边类型信息的情况下，模型的性能得到进一步提升，这表明节点类型信息和边类型信息能够提升编码模型对复杂程序的结构学习能力；不采用Transformer聚合编码层使得模型无法有效聚合前两层的信息，降低了模型捕获源代码的节点序列信息和语法结构信息的能力。

3.7　案例展示

为了进一步分析模型所生成的注释的质量，进行案例展示。Python示例和Java示例具体如下：

1）Python示例

```
def safe_decode(s):
if isinstance(s,unicode):
return s
elif isinstance(s,bytes):
return s.decode(defenc,'surrogateescape')
```

```
elif(s is not None):
raise TypeError('Expected bytes or text, but got %r' %(s,))
```

参考注释：safely decodes a binary string to unicode.

SiT：decodes a unicode string to unicode.

SHE：safely decodes a binary string to unicode.

2）Java示例

```
public String toString(){
return double.toString(get());
}
```

参考注释：returns the string representation of the current value.

SiT：obtains the string representation of this object.

SHE：returns the string representation of the double value.

由上述可以看出，SHE所生成的注释更准确和流畅。对于Python示例，虽然源代码中没有出现单词“binary”，但SHE所生成的注释仍然能够准确地预测出单词“binary”。对于Java示例，由于源代码中的方法名不足以描述该代码段的主要功能，因此SiT生成的注释的质量较差。但是，SHE生成的注释仍然具有较高的可读性和准确性。

4 结束语

本文针对代码注释生成任务，提出一种结构感知的混合编码（SHE）模型，通过融合序列编码层和图编码层，能够同时捕获程序代码的节点序列信息和语法结构信息。针对程序代码的语法类型信息，设计一种结构化感知的图注意力（SGAT）网络，提升了编码模型对于复杂程序代码的结构学习能力。实验通过包含多种编程语言的代码注释生成的数据集验证了SHE模型对比基准模型的提升效果，且能生成更准确的代码注释，同时消融实验也验证了SGAT网络的有效性。在未来的工作中，将进一步改进程序图的构建方式，使得程序图包含更多的语义信息，以对复杂程序代码有更好的结构学习能力。同时，将进一步优化序列编码层与图编码层的融合方式，以更好地捕获程序代码的序列信息和语法结构信息。

参考文献

[1] WOODFIELD S N, DUNSMORE H E, SHEN V Y. The effect of modularization and comments on program comprehension[C]//Proceedings of the 5th International Conference on Software Engineering. Washington D. C., USA: IEEE Press, 1981: 215-223.

[2] TENNY T. Program readability: procedures versus comments[J]. IEEE Transactions on Software Engineering, 1988, 14(9): 1271-1279.

[3] 殷明明，史小静，俞鸿飞，等. 基于对比注意力机制的跨语言句子摘要系统[J]. 计算机工程，2020，46(5)：86-93.
YIN M M, SHI X J, YU H F, et al. Cross-lingual sentence summarization system based on contrastive attention mechanism[J]. Computer Engineering, 2020, 46(5): 86-93. (in Chinese)

[4] 张楚婷，常亮，王文凯，等. 基于BiLSTM-CRF的细粒度知识图谱问答[J]. 计算机工程，2020，46(2)：41-47.
ZHANG C T, CHANG L, WANG W K, et al. Fine-grained question answering over knowledge graph based on BiLSTM-CRF[J]. Computer Engineering, 2020, 46(2): 41-47. (in Chinese)

[5] 冯读娟，杨璐，严建峰. 基于双编码器结构的文本自动摘要研究[J]. 计算机工程，2020，46(6)：60-64.
FENG D J, YANG L, YAN J F. Research on automatic text summarization based on dual-encoder structure[J]. Computer Engineering, 2020, 46(6): 60-64. (in Chinese)

[6] 陈翔，杨光，崔展齐，等. 代码注释自动生成方法综述[J]. 软件学报，2021，32(7)：2118-2141.
CHEN X, YANG G, CUI Z Q, et al. Survey of state-of-the-art automatic code comment generation[J]. Journal of Software, 2021, 32(7): 2118-2141. (in Chinese)

[7] 霍丽春，张丽萍. 代码注释演化及分类研究综述[J]. 内蒙古师范大学学报（自然科学汉文版），2020，49(5)：423-432.
HUO L C, ZHANG L P. An overview on evolution and classification of code annotation[J]. Journal of Inner Mongolia Normal University (Natural Sciences Edition), 2020, 49(5): 423-432. (in Chinese)

[8] 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. Stroudsburg, USA: Association for Computational Linguistics, 2016: 2073-2083.

[9] ALON U, BRODY S, LEVY O, et al. Code2Seq: generating sequences from structured representations of code[EB/OL]. [2021-11-05]. https://arxiv.org/abs/1808.01400.

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

[11] WU H Q, ZHAO H, ZHANG M. Code summarization with structure-induced Transformer[C]//Proceedings of ACL-IJCNLP'21. Stroudsburg, USA: Association for Computational Linguistics, 2021: 1078-1090.

[12] VASWANI A, SHAZEER N, PARMAR N, et al. Attention is all you need[C]//Proceedings of the 31st International Conference on Neural Information Processing Systems.

New York，USA：ACM Press，2017：5998-6008.

[13] LUONG M T，PHAM H，MANNING C D. Effective approaches to attention-based neural machine translation [EB/OL]. [2021-11-05]. https：//arxiv. org/abs/1508. 04025.

[14] HU X，LI G，XIA X，et al. Summarizing source code with transferred API knowledge[C]//Proceedings of the 27th International Joint Conference on Artificial Intelligence. Stockholm，Sweden：International Joint Conferences on Artificial Intelligence Organization，2018：2269-2275.

[15] LECLAIR A，JIANG S Y，MCMILLAN C. A neural model for generating natural language summaries of program subroutines [C]//Proceedings of the 41st International Conference on Software Engineering. Washington D. C.，USA：IEEE Press，2019：795-806.

[16] CHUNG J，GULCEHRE C，CHO K，et al. Empirical evaluation of gated recurrent neural networks on sequence modeling[EB/OL]. [2021-11-05]. https：//arxiv. org/pdf/1412. 3555. pdf.

[17] CAI R C，LIANG Z H，XU B Y，et al. TAG：type auxiliary guiding for code comment generation[C]//Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. Stroudsburg，USA：Association for Computational Linguistics，2020：291-301.

[18] FERNANDES P，ALLAMANIS M，BROCKSCHMIDT M. Structured neural summarization[EB/OL]. [2021-11-05]. https：//arxiv. org/abs/1811. 01824.

[19] LI Y J，TARLOW D，BROCKSCHMIDT M，et al. Gated graph sequence neural networks[EB/OL]. [2021-11-05]. https：//arxiv. org/abs/1511. 05493.

[20] LECLAIR A，HAQUE S，WU L F，et al. Improved code summarization via a graph neural network[C]//Proceedings of the 28th International Conference on Program Comprehension. New York，USA：ACM Press，2020：184-195.

[21] SCARSELLI F，GORI M，TSOI A C，et al. The graph neural network model[J]. IEEE Transactions on Neural Networks，2009，20(1)：61-80.

[22] LIU S Q，CHEN Y，XIE X F，et al. Retrieval-augmented generation for code summarization via hybrid GNN[EB/OL]. [2021-11-05]. https：//arxiv. org/pdf/2006. 05405. pdf.

[23] HELLENDOORN V J，SUTTON C，SINGH R，et al. Global relational models of source code[C]//Proceedings of the 8th International Conference on Learning Representations. New York，USA：ACM Press，2020：1-12.

[24] YASUNAGA M，LIANG P. Graph-based，self-supervised program repair from diagnostic feedback [EB/OL]. [2021-11-05]. https：//arxiv. org/abs/2005. 10636.

[25] 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. Stroudsburg，USA：Association for Computational Linguistics，2017：1073-1083.

[26] WAN Y，ZHAO Z，YANG M，et al. Improving automatic source code summarization via deep reinforcement learning[C]// Proceedings of the 33rd IEEE/ACM International Conference on Automated Software Engineering. Washington D. C.，USA：IEEE Press，2018：397-407.

[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 on Association for Computational Linguistics. Stroudsburg，USA：Association for Computational Linguistics，2002：311-318.

[28] LIN C Y. Rouge：a package for automatic evaluation of summaries [C]//Proceedings of Workshop on Text Summarization Branches Out. New York，USA：ACM Press，2004：74-81.

[29] BANERJEE S，LAVIE A. METEOR：an automatic metric for MT evaluation with improved correlation with human judgments[C]//Proceedings of Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization. New York，USA：ACM Press，2005：65-72.

[30] KINGMA D P，BA J. Adam：a method for stochastic optimization[EB/OL]. [2021-11-05]. https：//arxiv. org/pdf/1412. 6980. pdf.

[31] 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.

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

[33] 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. Stroudsburg，USA：Association for Computational Linguistics，2016：823-833.

[34] WEI B L，LI G，XIA X，et al. Code generation as a dual task of code summarization [C]//Proceedings of the 33rd International Conference on Neural Information Processing Systems. New York，USA：ACM Press，2019：6563-6573.

[35] AHMAD W，CHAKRABORTY S，RAY B，et al. A transformer-based approach for source code summarization[C]// Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. Stroudsburg，USA：Association for Computational Linguistics，2020：4998-5007.

编辑　陆燕菲