(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 115291854 A
(43)申请公布日 2022.11.04

(21)申请号 202210912454.0

(22)申请日 2022.07.30

(71)申请人 华为技术有限公司
地址 518129 广东省深圳市龙岗区坂田华为总部办公楼

(72)发明人 陈驰 张国安

(74)专利代理机构 北京亿腾知识产权代理事务所(普通合伙) 11309
专利代理师 刘辰雷 陈霁

(51)Int.Cl.
G06F 8/33(2018.01)
G06F 8/41(2018.01)
G06K 9/62(2022.01)
G06N 3/04(2006.01)
G06N 3/08(2006.01)

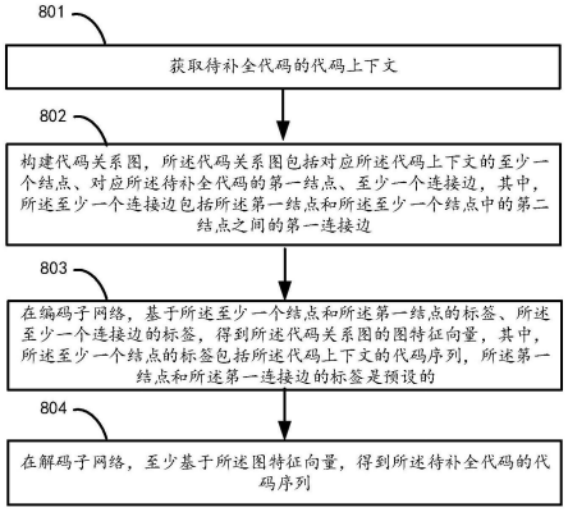
权利要求书3页 说明书20页 附图11页

(54)发明名称

代码补全方法、装置及设备

(57)摘要

本申请涉及计算机技术领域，具体涉及一种代码补全方法、装置及设备。该方法包括：获取待补全代码的代码上下文；构建代码关系图，代码关系图包括对应代码上下文的至少一个结点、对应待补全代码的第一结点、至少一个连接边，其中，至少一个连接边包括第一结点和至少一个结点中的第二结点之间的第一连接边；在编码子网络，基于至少一个结点和第一结点的标签、至少一个连接边的标签，得到代码关系图的图特征向量，其中，至少一个结点的标签包括代码上下文的代码序列，第一结点和第一连接边的标签是预设的；在解码子网络，至少基于图特征向量，得到待补全代码的代码序列。该方法具有较高的预测准确率。



CN 115291854 A

1.一种利用神经网络补全代码的方法,其特征在于,所述神经网络包括编码子网络和解码子网络,所述方法包括:

获取待补全代码的代码上下文;

构建代码关系图,所述代码关系图包括对应所述代码上下文的至少一个结点、对应所述待补全代码的第一结点、至少一个连接边,其中,所述至少一个连接边包括所述第一结点和所述至少一个结点中的第二结点之间的第一连接边;

在所述编码子网络,基于所述至少一个结点和所述第一结点的标签、所述至少一个连接边的标签,得到所述代码关系图的图特征向量,其中,所述至少一个结点的标签包括所述代码上下文的代码序列,所述第一结点和所述第一连接边的标签是预设的;

在所述解码子网络,至少基于所述图特征向量,得到所述待补全代码的代码序列。

2.根据权利要求1所述的方法,其特征在于,所述至少一个连接边还包括第二连接边,所述第二连接边连接所述至少一个结点中的第三结点和第四结点,所述第二连接边的标签为所述第三结点和所述第四结点之间的流关系,所述流关系为控制流和/或数据流。

3.根据权利要求1或2所述的方法,其特征在于,所述基于所述至少一个结点和所述第一结点的标签、所述至少一个连接边的标签,得到所述代码关系图的图特征向量,包括:

基于所述至少一个结点和所述第一结点的标签、所述至少一个连接边的标签,得到所述至少一个结点和所述第一结点的特征向量、以及所述至少一个连接边的特征向量;

将所述至少一个结点和所述第一结点的特征向量、以及所述至少一个连接边的特征向量,融合为所述图特征向量。

4.根据权利要求3所述的方法,其特征在于,所述基于所述至少一个结点和所述第一结点的标签、所述至少一个连接边的标签,得到所述至少一个结点和所述第一结点的特征向量、以及所述至少一个连接边的特征向量,包括:

基于所述第一结点的标签,得到所述第一结点的初始特征向量;基于所述第二结点的标签,得到所述第二结点的初始特征向量;以及基于所述第一连接边的标签,得到所述第一连接边的初始特征向量;

基于所述第一结点的初始特征向量和所述第一连接边的初始特征向量,得到所述第一结点的特征向量;以及基于所述第二结点的初始特征向量和所述第一连接边的初始特征向量,得到所述第二结点的特征向量;

基于所述第一结点的特征向量、所述第二结点特征向量和所述第一连接边的初始特征向量,得到所述第一连接边的特征向量。

5.根据权利要求4所述的方法,其特征在于,所述编码子网络包括第一层和第二层;其中,所述第一结点的初始特征向量和所述第二结点的初始特征向量是在所述第一层得到的,所述第一连接边的初始特征向量、所述第一结点的特征向量、所述第二结点的特征向量和所述第一连接边的特征向量是在所述第二层得到的。

6.根据权利要求1-5任一项所述的方法,其特征在于,所述至少一个结点的标签还包括所述代码上下文在抽象语法树中的类型。

7.根据权利要求1-6任一项所述的方法,其特征在于,所述编码子网络还包括第三层;

所述方法还包括:在所述第三层,基于所述代码上下文的文本信息,得到代码上下文的文本语义特征向量;所述文本信息包括所述代码上下文中的方法名、参数名、变量名、类名

中的至少一项；

所述至少基于所述图特征向量，得到所述待补全代码的代码序列包括：基于所述图特征向量和所述文本语义特征向量融合后的特征向量，得到所述待补全代码的代码序列。

8.根据权利要求1-7任一项所述的方法，其特征在于，所述第二结点对应的代码的位置与所述待补全代码的位置相邻。

9.根据权利要求1-8任一项所述的方法，其特征在于，所述代码序列为代码的标识符序列。

10.一种训练神经网络的方法，其特征在于，所述神经网络包括编码子网络和解码子网络，所述方法包括：

获取待补全代码的代码上下文；

构建代码关系图，所述代码关系图包括对应所述代码上下文的至少一个结点、对应所述待补全代码的第一结点、至少一个连接边，其中，所述至少一个连接边包括所述第一结点和所述至少一个结点中的第二结点之间的第一连接边；

在所述编码子网络，基于所述至少一个结点和所述第一结点的标签、所述至少一个连接边的标签，得到所述代码关系图的图特征向量，其中，所述至少一个结点的标签包括所述代码上下文的代码序列，所述第一结点和所述第一连接边的标签是预设的；

在所述解码子网络，至少基于所述图特征向量，得到所述待补全代码的预测代码序列；

基于所述预测代码序列和所述待补全代码的真实代码序列，得到预测损失；

朝着所述预测损失减少的方向，更新所述神经网络。

11.根据权利要求10所述的方法，其特征在，所述神经网络的参数量小于1亿。

12.一种利用神经网络代码补全的装置，其特征在于，所述神经网络包括编码子网络和解码子网络，所述装置包括：

获取单元，用于获取待补全代码的代码上下文；

构建单元，用于构建代码关系图，所述代码关系图包括对应所述代码上下文的至少一个结点、对应所述待补全代码的第一结点、至少一个连接边，其中，所述至少一个连接边包括所述第一结点和所述至少一个结点中的第二结点之间的第一连接边；

第一得到单元，用于在所述编码子网络，基于所述至少一个结点和所述第一结点的标签、所述至少一个连接边的标签，得到所述代码关系图的图特征向量，其中，所述至少一个结点的标签包括所述代码上下文的代码序列，所述第一结点和所述第一连接边的标签是预设的；

第二得到单元，用于在所述解码子网络，至少基于所述图特征向量，得到所述待补全代码的代码序列。

13.一种神经网络训练装置，其特征在于，所述神经网络包括编码子网络和解码子网络，所述装置包括：

获取单元，用于获取待补全代码的代码上下文；

构建单元，用于构建代码关系图，所述代码关系图包括对应所述代码上下文的至少一个结点、对应所述待补全代码的第一结点、至少一个连接边，其中，所述至少一个连接边包括所述第一结点和所述至少一个结点中的第二结点之间的第一连接边；

第一得到单元，用于在所述编码子网络，基于所述至少一个结点和所述第一结点的标

签、所述至少一个连接边的标签，得到所述代码关系图的图特征向量，其中，所述至少一个结点的标签包括所述代码上下文的代码序列，所述第一结点和所述第一连接边的标签是预设的；

第二得到单元，用于在所述解码子网络，至少基于所述图特征向量，得到所述待补全代码的预测代码序列；

第三得到单元，用于基于所述预测代码序列和所述待补全代码的真实代码序列，得到预测损失；

更新单元，用于朝着所述预测损失减少的方向，更新所述神经网络。

14.一种计算设备，其特征在于，包括处理器、存储器；所述存储器用于存储计算机程序；所述处理器用于执行所述计算机程序，以实现如权利要求1-10任一项所述的方法，或者如权利要求11所述方法。

15.一种计算机存储介质，其特征在于，所述计算机存储介质上存储有计算机程序，当所述计算机程序由处理器执行时，实现如权利要求1-10任一项所述的方法或者如权利要求11所述方法。

16.一种计算机程序产品，其特征在于，包括用于实现如权利要求1-10任一项所述方法或者如权利要求11所述方法的程序。

代码补全方法、装置及设备

技术领域

[0001] 本申请涉及计算机技术领域，具体涉及一种代码补全方法、装置及设备。

背景技术

[0002] 随着计算机技术的发展，计算机的功能日益丰富，因此，计算机程序开发愈发重要。程序员通常借助集成开发环境(integrated development environment，IDE)进行软件程序开发。IDE一般包括代码编辑器、编译器、调试器和图形用户界面等工具。其中，代码编辑器是用于程序员输入以及编辑代码的工具。代码编辑器提供代码补全功能，可以减少程序员输入的代码量，提升代码开发效率。其中，代码补全功能是指当程序员在代码编辑框敲入部分代码时，代码编辑器会提供预测出的至少一项后续部分代码。若该预测出的后续代码符合程序员的需要，则程序员会将预测出的后续代码补齐到代码编辑框中。

[0003] 目前，在软件开发领域，实现代码补全功能的主流方案有两种，分别为基于传统统计模型的方案以及基于深度学习的方案。其中，随着深度学习的发展，基于深度学习进行代码补全成为了软件开发领域的主要研究方向。

[0004] 无论使用哪种方案来实现代码补全与推荐，均需要更好地理解待补全代码的代码上下文，才能提高待补全代码的预测准确率。

发明内容

[0005] 本申请实施例提供了一种代码补全方法、装置及设备，可以提高待补全代码的预测准确率。

[0006] 第一方面，提供了一种利用神经网络补全代码的方法，神经网络包括编码子网络和解码子网络，该方法包括：获取待补全代码的代码上下文；构建代码关系图，代码关系图包括对应代码上下文的至少一个结点、对应待补全代码的第一结点、至少一个连接边，其中，至少一个连接边包括第一结点和至少一个结点中的第二结点之间的第一连接边；在编码子网络，基于至少一个结点和第一结点的标签、至少一个连接边的标签，得到代码关系图的图特征向量，其中，至少一个结点的标签包括代码上下文的代码序列，第一结点和第一连接边的标签是预设的；在解码子网络，至少基于图特征向量，得到待补全代码的代码序列。

[0007] 在该方法中，可以将代码上下文的代码序列，以及代码上下文和代码补全代码的之间的结构信息，融合到代码关系图中。在神经网络中，进一步融合代码序列和代码间的结构信息，得到代码关系图的图特征，从而实现了从多个角度刻画代码上下文和待补全代码之间的关联语义。根据代码关系图的图特征，预测待补全代码的代码序列，可以提升待补全代码的预测准确率。

[0008] 在一种可能的实现方式中，至少一个连接边还包括第二连接边，第二连接边连接至少一个结点中的第三结点和第四结点，第二连接边的标签为第三结点和第四结点之间的流关系，流关系为控制流和/或数据流。

[0009] 在该实现方式中，可以将代码上下文之间的结构信息，融合到代码关系图的图特

征中，使得在刻画代码上下文和待补全代码之间的关联语义时，还考虑到了代码上下文之间的结构信息，进而可以进一步提升待补全代码的预测准确率。

[0010] 在一种可能的实现方式中，基于至少一个结点和第一结点的标签、至少一个连接边的标签，得到代码关系图的图特征向量，包括：基于至少一个结点和第一结点的标签、至少一个连接边的标签，得到至少一个结点和第一结点的特征向量、以及至少一个连接边的特征向量；将至少一个结点和第一结点的特征向量、以及至少一个连接边的特征向量，融合为图特征向量。

[0011] 在该实现方式中，利用神经网络，可以先分别提取结点的特征信息和连接边的特征信息，然后，将结点的特征信息和连接边的特征信息，融合为图特征信息，从而使得图特征信息融合进了结点的代码序列以及结点之间的结构信息。由此，在基于图特征信息，预测代码序列时，可以得到较高的准确率。

[0012] 在一种可能的实现方式中，基于至少一个结点和第一结点的标签、至少一个连接边的标签，得到至少一个结点和第一结点的特征向量、以及至少一个连接边的特征向量，包括：基于第一结点的标签，得到第一结点的初始特征向量；基于第二结点的标签，得到第二结点的初始特征向量；以及基于第一连接边的标签，得到第一连接边的初始特征向量；基于第一结点的初始特征向量和第一连接边的初始特征向量，得到第一结点的特征向量；以及基于第二结点的初始特征向量和第一连接边的初始特征向量，得到第二结点的特征向量；基于第一结点的特征向量、第二结点特征向量和第一连接边的初始特征向量，得到第一连接边的特征向量。

[0013] 在该实现方式中，在提取结点的特征信息时，考虑连接边的特征信息，在提取连接边的特征信息时，考虑结点的特征信息，从而实现了代码的代码序列和代码间的结构信息的融合。

[0014] 在一种可能的实现方式中，编码子网络包括第一层和第二层；其中，第一结点的初始特征向量和第二结点的初始特征向量是在第一层得到的，第一连接边的初始特征向量、第一结点的特征向量、第二结点的特征向量和第一连接边的特征向量是在第二层得到的。其中，第一层可以为循环神经网络，第二层可以为图神经网络。

[0015] 在该实现方式中，可以在第一层提取代码序列的特征信息，以及在第二层提取代码间结构的特征信息，如此，可以在不牺牲预测准确率的情况下，减少神经网络的参数量。

[0016] 在一种可能的实现方式中，至少一个结点的标签还包括代码上下文在抽象语法树中的类型。

[0017] 在该实现方式中，可以将代码上下文在抽象语法树中的类型融合到代码关系图的图特征，从而实现了从更多个角度刻画代码上下文和待补全代码之间的关联语义，进而可以进一步提升待补全代码的预测准确率。

[0018] 在一种可能的实现方式中，编码子网络还包括第三层；该方法还包括：在第三层，基于代码上下文的文本信息，得到代码上下文的文本语义特征向量；文本信息包括代码上下文中的方法名、参数名、变量名、类名中的至少一项；至少基于图特征向量，得到待补全代码的代码序列包括：基于图特征向量和文本语义特征向量融合后的特征向量，得到待补全代码的代码序列。

[0019] 在该实现方式中，可以将代码上下文的文本语义特征和代码关系图的图特征进行

融合，并根据融合后的特征，预测待补全代码的代码序列，可以进一步提升待补全代码的预测准确率。

[0020] 在一种可能的实现方式中，第二结点对应的代码的位置与待补全代码的位置相邻。

[0021] 在该实现方式中，在待补全代码和其相邻的代码之间建立结构关系，并将该结构关系融合到图特征中，可以进一步提升待补全代码的预测准确率。

[0022] 在一种可能的实现方式中，代码序列为代码的标识符序列。

[0023] 代码的标识符可以反映代码的语义，将代码的标识符序列作为代码序列，参与刻画代码之间的关联语义，并据此预测待补全代码的代码序列，可以进一步提升待补全代码的预测准确率。

[0024] 第二方面，提供了一种训练神经网络的方法，神经网络包括编码子网络和解码子网络，该方法包括：获取待补全代码的代码上下文；构建代码关系图，代码关系图包括对应代码上下文的至少一个结点、对应待补全代码的第一结点、至少一个连接边，其中，至少一个连接边包括第一结点和至少一个结点中的第二结点之间的第一连接边；在编码子网络，基于至少一个结点和第一结点的标签、至少一个连接边的标签，得到代码关系图的图特征向量，其中，至少一个结点的标签包括代码上下文的代码序列，第一结点和第一连接边的标签是预设的；在解码子网络，至少基于图特征向量，得到待补全代码的预测代码序列；基于预测代码序列和待补全代码的真实代码序列，得到预测损失；朝着预测损失减少的方向，更新神经网络。

[0025] 在一种可能的实现方式中，神经网络的参数量小于1亿。

[0026] 在该实现方式中，所利用的神经网络的参数量较少，属于中小型模型，从而不经模型压缩，就可以将神经网络部署到代码编辑器侧，从而保障了待补全代码的预测准确率。

[0027] 第三方面，提供了一种利用神经网络补全代码的装置，神经网络包括编码子网络和解码子网络，该装置包括：获取单元，用于获取待补全代码的代码上下文；构建单元，用于构建代码关系图，代码关系图包括对应代码上下文的至少一个结点、对应待补全代码的第一结点、至少一个连接边，其中，至少一个连接边包括第一结点和至少一个结点中的第二结点之间的第一连接边；第一得到单元，用于在编码子网络，基于至少一个结点和第一结点的标签、至少一个连接边的标签，得到代码关系图的图特征向量，其中，至少一个结点的标签包括代码上下文的代码序列，第一结点和第一连接边的标签是预设的；第二得到单元，用于在解码子网络，至少基于图特征向量，得到待补全代码的代码序列。

[0028] 第四方面，提供了一种神经网络训练装置，神经网络包括编码子网络和解码子网络，该装置包括：获取单元，用于获取待补全代码的代码上下文；构建单元，用于构建代码关系图，代码关系图包括对应代码上下文的至少一个结点、对应待补全代码的第一结点、至少一个连接边，其中，至少一个连接边包括第一结点和至少一个结点中的第二结点之间的第一连接边；第一得到单元，用于在编码子网络，基于至少一个结点和第一结点的标签、至少一个连接边的标签，得到代码关系图的图特征向量，其中，至少一个结点的标签包括代码上下文的代码序列，第一结点和第一连接边的标签是预设的；第二得到单元，用于在解码子网络，至少基于图特征向量，得到待补全代码的预测代码序列；第三得到单元，用于基于预测代码序列和待补全代码的真实代码序列，得到预测损失；更新单元，用于朝着预测损失减少

的方向，更新神经网络。

[0029] 第五方面，提供了一种计算设备，包括处理器、存储器；存储器用于存储计算机程序；处理器用于执行计算机程序，以实现第一方面提供的方法，或者第二方面提供的方法。

[0030] 第六方面，提供了一种计算机存储介质，计算机存储介质上存储有计算机程序，当计算机程序由处理器执行时，实现如第一方面提供的方法，或者第二方面提供的方法。

[0031] 第七方面，提供了一种计算机程序产品，包括用于实现第一方面提供的方法，或者第二方面提供的方法的程序。

[0032] 本申请实施例提供的代码补充方法、装置及设备，可以将代码上下文的代码序列，以及代码上下文和代码补全代码的之间的结构信息，融合到代码关系图中。在神经网络中，进一步融合代码序列和代码间的结构信息，得到代码关系图的图特征，从而实现了从多个角度刻画代码上下文和待补全代码之间的关联语义。根据代码关系图的图特征，预测待补全代码的代码序列，可以提升待补全代码的预测准确率。

附图说明

[0033] 图1为本申请实施例提供的代码补全方案可应用的一种系统架构的结构示意图；

[0034] 图2为本申请实施例提供的一种代码补全模型训练方案的流程图；

[0035] 图3A为本申请实施例提供的一种代码关系图的示意图；

[0036] 图3B为本申请实施例提供的一种代码关系图的示意图；

[0037] 图4为本申请实施例提供的一种代码补全模型的结构示意图；

[0038] 图5为本申请实施例提供的一种代码补全方案的流程图；

[0039] 图6为本申请实施例提供的一种IDE的结构示意图；

[0040] 图7A为本申请实施例提供的一种IDE的结构示意图；

[0041] 图7B为本申请实施例提供的一种代码补全模型训练方案的流程图；

[0042] 图7C为本申请实施例提供的一种代码补全方案的流程图；

[0043] 图8为本申请实施例提供的一种代码补全方法的流程图；

[0044] 图9为本申请实施例提供的一种神经网络训练方法的流程图；

[0045] 图10为本申请实施例提供的一种利用神经网络补全代码的装置的结构示意图；

[0046] 图11为本申请实施例提供的一种神经网络训练装置的结构示意图；

[0047] 图12为本申请实施例提供的一种计算设备的结构示意图；

[0048] 图13为本申请实施例提供的一种计算设备的结构示意图。

具体实施方式

[0049] 下面将结合附图，对本申请实施例中的技术方案进行描述。显然，所描述的实施例仅是本申请一部分实施例，而不是全部的实施例。

[0050] 在本说明书的描述中"一个实施例"或"一些实施例"等意味着在本说明书的一个或多个实施例中包括结合该实施例描述的特定特征、结构或特点。由此，在本说明书中的不同之处出现的语句"在一个实施例中"、"在一些实施例中"、"在其他一些实施例中"、"在另外一些实施例中"等不是必然都参考相同的实施例，而是意味着"一个或多个但不是所有的实施例"，除非是以其他方式另外特别强调。

[0051] 其中，在本说明书的描述中，除非另有说明，"/"表示或的意思，例如，A/B可以表示A或B；本文中的"和/或"仅仅是一种描述关联对象的关联关系，表示可以存在三种关系，例如，A和/或B，可以表示：单独存在A，同时存在A和B，单独存在B这三种情况。另外，在本说明书实施例的描述中，"多个"是指两个或多于两个。

[0052] 在本说明书的描述中，术语"第一"、"第二"仅用于描述目的，而不能理解为指示或暗示相对重要性或者隐含指明所指示的技术特征的数量。由此，限定有"第一"、"第二"的特征可以明示或者隐含地包括一个或者更多个该特征。术语"包括"、"包含"、"具有"及它们的变形都意味着"包括但不限于"，除非是以其他方式另外特别强调。

[0053] 随着深度学习的发展，以及深度学习模型在自然语言处理领域取得的成功，利用深度学习模型进行代码补全已成为一个热门研究方向。利用深度学习模型理解待补全代码的代码上下文，并基于理解的结果，预测出待补全代码，从而减少程序员人工输入的代码量，提升软件开发效率。

[0054] 在一种方案中，将待补全代码的代码上下文解析为一维的文本序列，例如标识符(token)序列。然后，利用深度学习，根据文本序列，理解代码上下文的代码语义，并学习代码上下文和待补全代码之间的语义关联。进而根据语义关联，预测待补全代码。在该方案中，仅通过由文本序列刻画的代码语义，来学习不同代码之间的语义关联，而忽略了不同代码之间的结构信息。因此，该方案学习到的代码间的语义关联准确度较低，导致预测出的待补全代码的准确率也较低。

[0055] 另外，该方案采用的深度学习模型参数量较大，在亿级以上。该深度学习模型为了能够部署到代码编辑器侧，不得不采用知识蒸馏等技术压缩模型，这使得模型的预测准确率进一步降低。

[0056] 本申请实施例提供了一种利用神经网络补全代码的方案，可以采用代码关系图表示待补全代码的代码上下文以及待补全代码，代码关系图融合了代码的代码序列以及不同代码之间的结构关系等多种信息。在神经网络中，利用代码关系图可以从多个角度刻画不同代码之间的关联语义，进而可以基于该关联语义，预测待补全代码的代码序列。该方案提升了预测待补全代码的准确率。

[0057] 接下来，对申请实施例提供的补全代码方案进行具体介绍。

[0058] 首先，介绍本申请实施例可能涉及到的术语。

[0059] 标识符(token)：是程序员在编程时所使用的"名字"，具体是指代码进行分词后的最小单位，包括关键词、变量名、方法名、运算符、符号等。

[0060] 标识符序列：多个标识符按照一定顺序组成的序列。标识符序列为代码序列的一种表示形式。

[0061] 待补全位置：代码编辑框中需要输入代码但还没有输入代码的位置。待补全位置可以通过检测光标在代码编辑框中的位置得到。通常光标在代码编辑框中的位置为待补全位置。

[0062] 待补全代码：待补全位置需要输入的代码。待补全代码可以为标识符(token)或者标识符序列。

[0063] 代码补全：预测或者推荐待补全位置所需的待补全代码。

[0064] 代码上下文：代码上下文是指待补全位置或待补全代码的上下文代码，具体可以

是指待补全位置前后位置上的代码，或者待补全位置前面位置上的代码，或者待补全位置后面位置上的代码。

[0065] 函数(function)：可以直接被另一段程序或代码引用的程序或代码，可以实现一个或多个功能。一个函数由多个语句组成。

[0066] 控制流：同一函数中不同语句之间的执行顺序关联。

[0067] 数据流：同一函数中不同语句之间的数据依赖关联。

[0068] 抽象语法树(abstract syntax tree，AST)：代码的抽象语法结构，表现为树状。换言之，抽象语法树以树状的形式表现函数的语法结构，树上的每个结点都表示函数中的一种结构。

[0069] 其次，介绍申请实施例提供的补全代码方案可应用的系统架构。如图1所示，该系统架构可以包括用于训练代码补全模型120的代码库111、代码补全模型的代码解析器112、代码表示构造器113以及训练数据构造器114，以及包括用于为代码编辑器130补全代码的代码解析器141、代码表示构造器142。

[0070] 其中，代码补全模型的代码解析器112、代码表示构造器113以及训练数据构造器114、代码解析器141、代码表示构造器142为具有数据处理能力的功能模块。在一些实施例中，代码补全模型的代码解析器112、代码表示构造器113以及训练数据构造器114、代码解析器141或代码表示构造器142，可以通过软件方式实现，例如可以为进程(process)、虚拟计算实例等。其中，虚拟计算实例可以为虚拟机(virtual machine，VM)或者容器(container)。本申请实施例对这些功能模块的具体实现方式不做限定。

[0071] 代码补全模型120为一种基于深度学习的神经网络，其具体结构将在下文进行介绍，在此不再赘述。

[0072] 代码编辑器130为用于用户(例如程序员)输入以及编辑代码的工具。示例性的，代码编辑器130可以为集成到IDE中，而也可以单独实现。代码编辑器130可以向用户提供代码编辑框，使得用户可以在代码编辑框中输入或编辑代码。

[0073] 接下来，结合图1所示的系统架构，示例介绍本申请实施例提供的代码补全方案。其中，该代码补全方案可分为代码补全模型120的训练过程和推理过程。

[0074] 首先，介绍代码补全模型120的训练过程。

[0075] 参阅图2，代码解析器112可以执行步骤201，可以从代码库111中获取源代码。其中，获取到的源代码可以包括至少一个函数。在一些实施例中，可以从代码库111获取至少一个文件的源代码，一个文件的源代码可以包括至少一个函数。在一个例子中，源代码可以为基于Python语言编写的代码。在另一个例子中，源代码可以为基于TypeScript语言编写的代码。在又一个例子中，源代码可以为基于Java语言编写的代码。前文仅对从代码库111中获取源代码的方式以及源代码的语言进行示例说明，并非限定。在其他例子中，还可以采用其他方式从代码库111中获取源代码，或者源代码可以为基于其他语言编写的代码，在此不再一一赘述。

[0076] 代码解析器112可以执行步骤202，解析源代码。在一些实施例中，从代码库111获取的源代码包括至少一个文件的代码。则可以以文件为单位，将每个文件中的源代码解析为抽象语法树。其中，一个文件的源代码可以包括至少一个函数。在将源代码解析为抽象语法树后，可以提取每个函数对应的抽象语法树。由此，完成源代码解析。

[0077] 代码解析器112可以执行步骤203，将解析后的源代码发送至代码表示构造器113。代码表示构造器113可以执行步骤204，基于解析后的源代码构建代码关系图。其中，可以以函数为单位，构建代码关系图，即一个函数，可以构建为一个代码关系图。具体而言，可以基于函数对应的抽象语法树，对函数所包括的多个语句进行程序分析。其中，程序分析包括语句类型解析、流关系分析。其中，流关系包括控制流和数据流。其中，两个结点之间的流关系可以为控制流或数据流，也可以为控制流和数据流。控制流表示两个结点之间有直接控制流关联。数据流表示两个结点之间有直接数据流关联。控制流与数据流表示两个结点之间既有直接控制流又有直接数据流关联。流关系分析是指分析两个语句之间是否存在控制流或/或数据流。经过程序分析，构建包括结点和连接边的代码关系图。

[0078] 参阅图3A，以基于函数A，构建的代码关系图A1为例，代码关系图A1包括了多个结点，例如结点a11、结点a12、结点a13、结点a14、结点a15等。其中，每个结点对应函数A中的一条语句。在一个例子中，当一条语句的代码序列的长度超过阈值Y1时，该条语句可以被分成至少两个子语句，其中，每个子语句对应一个结点。该至少两个子语句中相邻子语句对应的结点之间具有追加控制流。其中，追加控制流属于控制流，代表了该至少两个子语句之间的执行顺序。

[0079] 代码关系图A1中结点具有标签。其中，标签为结点的信息，包括了结点的特征。具体而言，结点的标签包括了结点对应语句的代码序列。在一些实施例中，代码序列可以由标识符序列表示。其中，可以对语句的代码序列进行分词，得到该语句的标识符序列。在一些实施例中，可以采用自定义分词方法，对语句进行分词，得到该语句的标识符序列。在一些实施例中，可以采用bpe分词法，对语句进行分词，得到该语句的标识符序列。在一些实施例中，可以采用自定义分词结合bpe分词的方法，对语句进行分词，得到该语句的标识符序列。具体而言，先按照空格对代码语句的原始标识符序列进行切分，得到初切分标识符序列；其次，按照标点符号对初切分标识符序列进行切分，得到再切分标识符序列；最后，按照驼峰命名法、snake命名法以及数字对再切分标识符序列进行切分，得到三次切分标识符序列；按照bpe分词，对三次切分标识符序列进行切分，得到最终的分词结果。可以将最终得分词结果作为对应结点的标签。

[0080] 在一些实施例中，代码关系图A1中结点的标签还包括该结点对应的语句在抽象语法树中的类型。

[0081] 继续参阅图3A，具有流关系的结点之间具有连接边，该连接边的标签为结点间的流关系。例如，结点a11和结点a12之间具有控制流，则结点a11和结点a12之间的连接边的标签为控制流。再例如，结点a12和结点a13之间具有数据流和控制流，则结点a12和结点a13之间的连接边的标签为数据流和控制流。

[0082] 在一些实施例中，对于选择语句，循环语句，异常处理语句等控制结构语句，会增加Condition结点、Then结点、Body结点、OutControl结点等作为占位符结点来表征控制结构语句的结构信息。前述占位符结点会按序分别作为控制结点的孩子结点添加到代码关系图表示上。其中，控制结点为对应于控制结构语句的结点。Condition结点的孩子结点为控制结构语句的条件、Then结点和/或Body结点的孩子结点为控制结构语句的结构体中的代码、OutControl结点的孩子结点为控制结构语句结构体外的代码。

[0083] 在一些实施例中，代码关系图中的结点还可以携带对应该结点的语句的文本信

息。在一些实施例中，语句的文本信息可以包括该语句的类名、方法名、参数名、变量名中的至少一种。其中，类名是语句所属类的名字，方法名是语句所属函数的名字，参数名是语句中参数的名字，变量名是语句中变量的名字，这些名字的本身包含了语句语义，将它们作为文本信息，用于提取语句的文本语义特征，来刻画语句的语义，可以提高预测待补全代码的准确率。

[0084] 如此，代码关系图融合了不同的代码序列以及不同语句之间的结构关系等多种信息，从而可以用于在后续步骤中，从多个角度刻画不同代码之间的关联语义。

[0085] 继续参阅图2，训练数据构造器114可以通过步骤205，从代码表示构造器113获取代码关系图A1，并执行步骤206，生成代码关系图A11。其中，代码关系图A11为训练数据，用于训练代码补全模型120。具体而言，在步骤206中，可以将代码关系图A1中的部分结点移除。其中，被移除结点的代码序列可以作为真值(ground truth)，用于后续对代码补全模型120进行有监督训练。在移除了结点的位置添加窟窿(hole)结点。窟窿结点代表或者说对应待补全代码。被移除结点的代码序列为窟窿结点的真实代码序列。其中，窟窿结点的标签设置为预设的标签，为方便描述，可以将该预设的标签称为窟窿标签。其中，窟窿标签可以理解为占位符标签，其不含代码序列，也不含语句的类型。在一些实施例中，窟窿标签可以为预设个数的占位符，例如20个占位符、15个占位符、10个占位符或者5个占位符等。

[0086] 在结点被移除后，创建与被移除结点具有流关系的结点和窟窿结点之间的连接边，该连接边代表了特殊流。其中，特殊流是指未知流关系的流。代表特殊流的连接边具有预设的标签，为方便描述，可以将该预设的标签称为特殊流标签。特殊流标签可以理解为占位符标签。在一些实施例中，特殊流标签可以为预设个数的占位符，例如10个占位符、5个占位符或者2个占位符等。

[0087] 在一些实施例中，可以通过如下方式构造代码关系图A11。

[0088] 首先，迭代地从代码关系图A1的根结点的孩子结点开始进行遍历，移除与当前遍历到的结点存在控制流关联的后续若干结点(若干结点是一个可以自定义的阈值范围是1～结点总数-1)。按照控制流关联移除结点，可以保障代码结构的语法正确性。其次，去除与所有移除掉的结点有关联的所有边。再次，将所有移除掉的结点用一个窟窿结点替代，并且与窟窿结点相连的结点之间的连接边的类型改为特殊流。设定在代码关系图A1中移除结点a12，则可以得到如图3B所示的代码关系图A11。其中，结点a12所在的代码关系图中可以替换为窟窿结点，窟窿结点和结点A11之间的连接边代表特殊流，窟窿结点和结点a13之间的连接边代表特殊流。

[0089] 可以将移除掉结点中的第一个结点的代码序列作为真值(ground truth)，可以得到用于进行监督训练的训练数据，即代码关系图A11。即将移除掉结点中的第一个结点用窟窿结点代替，然后，预测窟窿结点的代码序列，并且预测到的代码序列和移除掉结点中的第一个结点的代码序列进行对比，得到预测损失。其中，预测损失也可以称为损失函数。由此，可以在朝着预测损失减少的方向，更新代码补全模型120，以进行代码补全模型120的训练。

[0090] 在一些实施例中，可以将未被移除的结点对应的语句的文本信息，进行汇总，并去重，得到代码关系图A11的文本信息。其中，如上所述，语句的文本信息包括语句的类名、方法名、参数名、变量名等中的至少一种。当对语句的类名、方法名、参数名、变量名不做特别区分时，它们可以被简称为词或者文本词。相应地，代码关系图A11的文本信息也可以称为

代码关系图A11的词袋或者文本词袋。

[0091] 如图4所示,代码补全模型120可以包括编码子网络和解码子网络,其中,编码子网络用于提取代码关系图的图特征,解码子网络用于根据代码关系图的图特征,预测待补全代码的代码序列。接下来,对编码子网络和解码子网络分别进行介绍。

[0092] 其中,代码补全模型120的训练可以由训练模块执行。在一些实施例中,训练模块可以为物理设备,例如服务器。在一些实施例中,训练模块可以为虚拟计算实例。本申请实施例对训练模块的具体实现形式不做限定。

[0093] 训练模块可以执行步骤207,从训练数据构造器114获取代码关系图A11,并将代码关系图A11输入到代码补全模型120。其中,训练模块可以在编码子网络,执行步骤208,提取代码关系图A11的图特征。其中,图特征可以通过特征向量表示。在步骤208中,可以基于代码关系图上各个结点的标签以及各个连接边的标签,得到该代码关系图的图特征向量。其中,如图4所示,编码子网络包括嵌入层。在嵌入层,可以对代码关系图进行图嵌入处理,得到嵌入向量。其中,得到的嵌入向量可以包括结点的标签的嵌入向量。然后,可以基于嵌入向量,提取代码关系图的图特征向量。

[0094] 在一些实施例中,在编码子网络中,基于代码关系图上各个结点的标签以及各个连接边的标签,得到各个结点的特征向量和各个连接边的特征向量。然后,将各个结点的特征向量和各个连接边的特征向量融合为代码关系图的特征向量。在一些实施例中,融合可以通过求和、取平均、最大池化、注意力机制中的一种或多种实现。

[0095] 在该实施例的一个说明性示例中,如图4所示,编码子网络可以包括层C1和层C2。其中,在层C1,可以基于代码关系图中每个结点的标签,得到该结点的初始特征向量。然后,将每个得到的初始特征向量输入到层C2。在层C2,可以基于代码关系图中每个连接边的标签,得到该连接边的初始特征向量。在层C2,可以根据该连接边的初始特征向量和该连接边所连接的结点的初始特征向量,得到该连接边所连接的结点的特征向量。之后,可以根据连接边两端所连接的结点的特征向量和该连接边的初始特征向量,得到该连接边的特征向量。

[0096] 在一些实施例中,层C1的结构可以为循环神经网络(rerrent neural network,RNN)。在一个例子中,层C1的结构可以为常规RNN、长短期记忆网络(long short-term memory,LSTM)、门控循环单元(gated recurrent unit,GRU)等中的任一项。在其他例子中,层C1的结构可以为其他形式的循环神经网络。

[0097] 在一些实施例中,层C2的结构可以为图神经网络(graph neural network,GNN)。在一个例子中,层C2的结构可以为门控图神经网络(gated graph neural network,GGNN)、图注意力网络(Graph attention network,GAT)、图卷积网络(graph convolutional network,GCN)等中的任一项。在其他例子中,层C2的结构可以为其他形式的图神经网络。

[0098] 在一些实施例中,如图4所示,编码子网络还包括层C3。如上所述,代码关系图中的结点还可以携带对应该结点的语句的文本信息。可以将代码关系图的文本信息输入到层C3。具体地,可以对文本信息中每个文本词进行嵌入处理,得到词嵌入向量,并将词嵌入向量输入到层C3。在层C3,可以根据每个词嵌入向量,得到词特征向量。然后,将文本信息中各个词的词特征向量,进行融合,得到图的文本语义特征向量。在一些实施例中,融合可以通过求和、取平均、最大池化、注意力机制中的一种或多种实现。

[0099] 在编码子网络的层C4,将代码关系图的文本语义特征向量和代码关系图的图特征向量进行融合,得到代码关系图的融合特征向量。其中,融合可以通过拼接、求和、全连接中的一种或多种实现。

[0100] 接下来,可以将代码关系图的图特征向量或者代码关系图的融合特征向量,输入到解码子网络。在解码子网络可以执行步骤209,基于图特征,预测待补全代码的代码序列。具体而言,可以基于代码关系图的图特征向量或者代码关系图的融合特征向量,预测待补全代码的代码序列。具体如下。

[0101] 如图4所示,解码子网络可以包括分类层。在分类层可以基于代码关系图的图特征向量或者代码关系图的融合特征向量,预测待补全代码的代码序列。在一些实施例中,在分类层,可以通过softmax机制,基于代码关系图的图特征向量或者代码关系图的融合特征向量,预测该代码关系图中待补全代码的代码序列

[0102] 在一些实施例中,如图4所示,在分类层的数据输入侧还包括层C5,其中,在层C5,可以从代码关系图的图特征向量或者代码关系图的融合特征向量中,进一步学习代码关系图中各个结点间的关联语义,并将学习结果传递至分类层,使得分类层可以根据学习结果,预测待补全代码的代码序列,提高预测的准确率。

[0103] 在一些实施例中,如图4所示,解码子网络可以包括N个层集合,每个层集合包括层C5和分类层组成。其中,N为大于或等于1的整数。该N个层集合按照时间序列展开。其中,一个层集合中的C5可以从代码关系图的图特征向量或者代码关系图的融合特征向量中,或者从前一个层集合的分类层输出的预测结果中,进一步学习代码关系图中各个结点间的关联语义,并将学习结果传递至分类层,使得分类层可以根据学习结果,预测待补全代码的代码序列,提高预测的准确率。

[0104] 在一些实施例中,层C5为循环神经网络。在一个例子中,层C5的结构可以为RNN、LSTM、GRU等中的任一项。在其他例子中,层C5的结构可以为其他形式的循环神经网络。

[0105] 训练模块可以执行步骤210,将预测的待补全代码的代码序列和该待补全代码对应的窟窿结点的真值进行比较,得到损失函数。然后,训练模块在步骤211中,朝着损失函数减少的方向,训练代码补全模型120。其中,预测的待补全代码的代码序列也可以称为待补全代码的预测代码序列,该待补全代码对应的窟窿结点的真值是指待补全代码的真实代码序列。

[0106] 在一些实施例中,代码补全模型120为中小型深度学习模型,其参数量小于1亿。

[0107] 上文示例介绍了代码补全模型120的训练过程。接下来,介绍代码补全模型120的推理过程。

[0108] 参阅图5,代码解析器141可以执行步骤501,从代码编辑器130获取代码上下文。其中,代码上下文为待补全代码的代码上下文,即待补全位置的代码上下文。如上所述,作为训练数据的代码关系图是以函数为单位进行构建。因此,获取的待补全代码的代码上下文属于同一函数,相应地,待补全代码也应与该代码上下文属于同一函数。

[0109] 代码编辑器130可以提供代码编辑框。代码编辑框用于输入和编辑代码。代码编辑框的光标所在位置,为当前要输入或编辑代码的位置。因此,可以检测光标在代码编辑框的位置,并将检测到的位置作为待补全位置,需要在待补全位置输入的代码为待补全代码。

[0110] 在一些实施例中,参阅图6,可以在代码编辑器130配置代码补全接口131。该代码

补全接口131用于检测光标在代码编辑框中的位置，从而得到待补全位置，进而得到待补全代码的代码上下文。该代码补全接口131可以将代码上下文发送至代码解析器141。该代码补全接口还可以提供(例如显示)预测出的待补全代码的序列，并接收用户的选择操作，以及根据用户的选择操作，将预测出的待补全代码的序列输入到待补全位置，完成代码补全。

[0111] 在一些实施例中，如图6所示，代码解析器141可以集成到代码编辑器130所在的IDE中，例如以插件形式被加载到IDE中。即代码解析器141可以设置在代码编辑器130的本地。

[0112] 代码解析器141可以执行步骤502，解析代码上下文。具体而言，可以将该代码上下文所在的函数解析为抽象语法树。

[0113] 代码解析器141可以执行步骤503，将解析后的代码上下文发送至代码表示构造器142。代码表示构造器142可以执行步骤504，基于解析后的代码上下文构建代码关系图A2。具体而言，可以基于函数对应的抽象语法树，对代码上下文所包括的语句进行程序分析。其中，用户在开发代码时，函数的函数头是必须输入的，因此，代码上下文至少包括代码上下文所在函数的函数头。函数的函数头为一个语句，因此，代码上下文包括至少一个语句。当然，代码上下文还可以包括更多个语句。代码上下文所包括的语句的数量可由代码编辑框中已输入语句的数量确定。

[0114] 对语句的程序分析包括语句类型解析、流关系分析。其中，流关系包括控制流和数据流。其中，两个结点之间的流关系可以为控制流或数据流，也可以为控制流和数据流。控制流表示两个结点之间有直接控制流关联。数据流表示两个结点之间有直接数据流关联。控制流与数据流表示两个结点之间既有直接控制流又有直接数据流关联。流关系分析是指分析两个语句之间是否存在控制流或/或数据流。

[0115] 经过程序分析，构建包括至少一个结点的代码关系图。该至少一个结点对应代码上下文。其中，当代码上下文中有多个语句时，该至少一个结点可以包括多个结点。该多个结点中的每个结点对应多个语句中的一个语句，且不同的语句对应的结点不同。另外，该多个结点中具有流关系的结点之间具有连接边，该连接边的标签为该流关系。也就是说，连接边的标签是该连接边所连接的两个结点之间的流关系。其中，流关系可以为控制流和/或数据流。也就是说，流关系可以在控制流、数据流中的选择。两个结点之间的流关系具体是哪种流，是由两个结点的流关系分析所确定的。

[0116] 该至少一个结点(即对应代码上下文的结点)的标签包括代码上下文的代码序列。其中，该至少一个结点中的一个结点对应代码上下文中的一个语句，该结点的标签包括对应该结点的语句的代码序列。在一些实施例中，代码序列可以由标识符序列表示。其中，可以对语句或者代码上下文的代码序列进行分词，得到该语句的标识符序列。具体可以参考上文对步骤204的介绍实现，在此不再赘述。

[0117] 在一些实施例中，该至少一个结点(即对应代码上下文的结点)还可以包括对应的代码上下文在抽象语法树中的类型。具体而言，该至少一个结点中每个结点的标签还包括该结点对应的语句在抽象语法树中的类型。其中该抽象语法树为该代码上下文所属函数的抽象语法树。具体参考可以上文对代码关系图A1的介绍，在此不再赘述。

[0118] 在一些实施例中，该至少一个结点(即对应代码上下文的结点)还可以携带代码上下文的文本信息。其中，该至少一个结点中的一个结点对应代码上下文中的一个语句，该结

点还可以携带对应该结点的语句的文本信息。在一些实施例中，语句的文本信息可以包括该语句的类名、方法名、参数名、变量名中的至少一种。其中，类名是语句所属类的名字，方法名是语句所属函数的名字，参数名是语句中参数的名字，变量名是语句中变量的名字，这些名字的本身包含了语句语义，将它们作为文本信息，用于提取语句的文本语义特征，来刻画语句的语义，可以提高预测待补全代码的准确率。

[0119] 在步骤504中，可以将待补全位置作为一个窟窿结点，添加到代码关系图中，得到代码关系图A2。其中，窟窿结点代表或者说对应待补全代码。其中，窟窿结点的标签设置为窟窿标签。其中，窟窿标签可以参考上文对窟窿标签的介绍实现，在此不再赘述。

[0120] 可以设定至少一个结点包括结点B1，可以构建窟窿结点和至少一个结点(即对应代码上下文的结点)之间的连接边B11，即代码关系图A2包括窟窿结点和结点B1之间的连接边B11。其中，连接边B11代表特殊流，连接边B11的标签为特殊流标签。特殊流标签可以参考上文对特殊流标签的介绍实现，在此不再赘述。

[0121] 其中，结点B1对应的语句的位置与待补全代码的位置相邻。也就是说，结点B1对应的语句的位置与待补全位置相邻。其中，位置是指在代码编辑框中的位置。在一些实施例中，结点B1对应的语句的位置与待补全位置相邻，且位于待补全位置之前。在一些实施例中，结点B1对应的语句的位置与待补全位置相邻，且位于待补全位置之后。在一些实施例中，结点B1可以为两个结点，该两个结点中一个结点对应的语句与待补全位置相邻，且位于待补全位置之前，另一个结点对应的语句与待补全位置相邻，且位于待补全位置之后。其中，当结点B1为两个结点时，这两个结点中的一个结点与窟窿结点之间具有连接边，另一个结点与窟窿结点之间具有连接边。且这两个连接边均带有特殊流标签。

[0122] 经过上述步骤，代码表示构造器142可以执行步骤505，将代码关系图A2发送至推理模块143。其中，推理模块143也可以称为模型推理服务。推理模块143配置有代码补全模型120。其中，此处的代码补全模型120为经过图2所示训练方案训练后的模型。

[0123] 在一些实施例中，推理模块143可以通过软件方式实现，例如可以为进程(process)、虚拟计算实例等。其中，虚拟计算实例可以为虚拟机。本申请实施例对这些功能模块的具体实现方式不做限定。

[0124] 在一些实施例中，如图6所示，代码解析器141、代码表示构造器142、推理模块143可以作为IDE组件，配置在于代码编辑器120所在的IDE中。其中，如上所述，代码补全模型120为中小型模型，其参数量小于1个亿，因此，代码补全模型120可以不经压缩，就可以配置在推理模块143中，如此，可以保障了代码补全模型120的预测准确率。

[0125] 推理模块143可以利用代码补全模型120，执行代码补全方案，即预测代码关系图A2中窟窿结点的代码序列。

[0126] 接下来，结合图4和图5，对代码补全方案进行具体介绍。

[0127] 如图4所示，代码补全模型120可以包括编码子网络和解码子网络，其中，编码子网络用于提取代码关系图的图特征，解码子网络用于根据代码关系图的图特征，预测待补全代码的代码序列。接下来，对编码子网络和解码子网络分别进行介绍。

[0128] 如图5所示，推理模块143可以执行步骤506，利用代码补全模型120，提取代码关系图A2的图特征。具体而言，推理模块143可以将代码关系图A2输入到代码补全模型120。其中，在代码补全模型120的编码子网络，执行步骤506，提取代码关系图A2的图特征。其中，图

特征可以通过特征向量表示。在步骤506中，可以基于代码关系图A2上各个结点的标签以及各个连接边的标签，得到该代码关系图的图特征向量。其中，如图4所示，编码子网络包括嵌入层。在嵌入层，可以对代码关系图A2进行图嵌入处理，得到嵌入向量。其中，得到的嵌入向量可以包括结点的标签的嵌入向量。然后，可以基于嵌入向量，提取代码关系图的图特征向量。

[0129] 在一些实施例中，在编码子网络中，基于代码关系图A2上各个结点的标签以及各个连接边的标签，得到各个结点的特征向量和各个连接边的特征向量。然后，将各个结点的特征向量和各个连接边的特征向量融合为代码关系图的特征向量。在一些实施例中，融合可以通过求和、取平均、最大池化、注意力机制中的一种或多种实现。

[0130] 在该实施例的一个说明性示例中，如图4所示，编码子网络可以包括层C1和层C2。其中，在层C1，可以基于代码关系图A2中每个结点的标签，得到该结点的初始特征向量。然后，将每个结点的初始特征向量输入到层C2。其中，连接边的标签可以经过层C1输入到层C2。在层C2，可以基于代码关系图中每个连接边的标签，得到该连接边的初始特征向量。在层C2，可以根据该连接边的初始特征向量和该连接边所连接的结点的初始特征向量，得到该连接边所连接的结点的特征向量。之后，可以根据连接边两端所连接的结点的特征向量和该连接边的初始特征向量，得到该连接边的特征向量。

[0131] 在一些实施例中，层C1的结构可以为循环神经网络。在一个例子中，层C1的结构可以为常规RNN、LSTM、GRU等中的任一项。在其他例子中，层C1的结构可以为其他形式的循环神经网络。

[0132] 在一些实施例中，层C2的结构可以为图神经网络。在一个例子中，层C2的结构可以为GGNN、GAT、GCN等中的任一项。在其他例子中，层C2的结构可以为其他形式的图形神经网络。

[0133] 在一些实施例中，如图4所示，编码子网络还包括层C3。如上所述，代码关系图A2中的结点还可以携带对应该结点的语句的文本信息。可以将代码关系图A2的文本信息输入到层C3。具体地，可以对文本信息中每个文本词进行嵌入处理，得到词嵌入向量，并将词嵌入向量输入到层C3。在层C3，可以根据每个词嵌入向量，得到词特征向量。然后，将文本信息中各个词的词特征向量，进行融合，得到代码关系图A2的文本语义特征向量。在一些实施例中，融合可以通过求和、取平均、最大池化、注意力机制中的一种或多种实现。

[0134] 在编码子网络的层C4，将代码关系图A2的文本语义特征向量和代码关系图A2的图特征向量进行融合，得到代码关系图A2的融合特征向量。其中，融合可以通过拼接、求和、全连接中的一种或多种实现。

[0135] 接下来，可以将代码关系图A2的图特征向量或者代码关系图A2的融合特征向量，输入到解码子网络。推理模块142在解码子网络可以执行步骤507，基于代码关系图的图特征，预测待补全代码的代码序列。具体而言，可以基于代码关系图A2的图特征向量或者代码关系图的融合特征向量，预测待补全代码的代码序列。具体如下。

[0136] 如图4所示，解码子网络可以包括分类层。在分类层可以基于代码关系图A2的图特征向量或者代码关系图A2的融合特征向量，预测待补全代码的代码序列。在一些实施例中，在分类层，可以通过softmax机制，基于代码关系图A2的图特征向量或者代码关系图的融合特征向量，预测该代码关系图中待补全代码的代码序列

[0137] 在一些实施例中，如图4所示，在分类层的数据输入侧还包括层C5，其中，在层C5，可以从代码关系图A2的图特征向量或者代码关系图A2的融合特征向量中，进一步学习代码关系图中各个结点间的关联语义，并将学习结果传递至分类层，使得分类层可以根据学习结果，预测待补全代码的代码序列，提高预测的准确率。

[0138] 在一些实施例中，如图4所示，解码子网络可以包括N个层集合，每个层集合包括层C5和分类层组成。其中，N为大于或等于1的整数。该N个层集合按照时间序列展开。其中，一个层集合中的C5可以从代码关系图A2的图特征向量或者代码关系图A2的融合特征向量中，或者从前一个层集合的分类层输出的预测结果中，进一步学习代码关系图中各个结点间的关联语义，并将学习结果传递至分类层，使得分类层可以根据学习结果，预测待补全代码的代码序列，提高预测的准确率。

[0139] 在一些实施例中，层C5为循环神经网络。在一个例子中，层C5的结构可以为RNN、LSTM、GRU等中的任一项。在其他例子中，层C5的结构可以为其他形式的循环神经网络。

[0140] 如此，经过步骤507，可以预测出待补全代码的代码序列。

[0141] 在一些实施例中，推理模块143还可以执行步骤508，将预测出的待补全代码的代码序列发送至代码编辑器130。

[0142] 在一些实施例中，在步骤507，解码子网络可以预测出多个代码序列，其中，该多个代码序列的概率大小不一。其中，代码序列的概率是指该代码序列为待补全代码的正确代码序列的可能性。推理模块143可以按照该多个代码序列的概率的大小，对该多个代码序列的进行排序，得到排序结果。示例性的，推理模块143可以通过束搜索(beam search)算法，按照该多个代码序列的概率的大小，对该多个代码序列的进行排序，得到排序结果。在步骤508中，推理模块143可以将该排序结果，发送至508.

[0143] 代码编辑器130可以执行步骤509，提供预测出的待补全代码的代码序列。在一些实施例中，代码编辑器130可以在代码编辑框中待补全位置处或者附近，显示预测出的待补全代码的代码序列，以便用户选择或者确认预测出的待补全代码的代码序列。在一个示例中，预测出的待补全代码的代码序列可以包括多个代码序列，代码编辑器130可以在下拉选择框中显示该多个代码序列，以便用户从该多个代码序列中选择代码序列。其中，用户选择的代码序列用作待补全代码的最终代码序列或者说正确代码序列。

[0144] 本申请实施例提供的代码补全方案，采用了多种信息，例如代码的代码序列以及不同代码之间的结构关系(即流关系)，以及文本语义信息等，来刻画代码之间的关联语义。并且，本申请实施例将该多元信息融合在代码关系图上，并提取代码关系图的图特征，从而将多种信息的特征进行了融合，然后，利用融合后的特征，预测待补全代码的代码序列，提升了预测准确率。

[0145] 另外，本申请实施例训练出的代码补全模型为中小型模型，可以不经模型压缩，就可以部署到代码编辑器侧，这进一步保障了预测准确率。

[0146] 接下来，在实施例1中，以基于Python语言编写的代码为例，举例介绍本申请实施例提供的方案。具体如下，

[0147] 在PyCharm IDE中。当用户在PyCharm的代码编辑器的function(即所写好的python function)内部键入字符后，触发python代码token补全动作，利用用户本地机器的计算资源，调用模型进行推理服务，最终返回结果给用户。

[0148] 图7A示出了实施例1的组件结构图。在PyCharm中，本实施例涉及到的模块包括代码token补全入口、Python PSI、Python代码表示构造器和Python模型推理服务。其中，代码token补全入口也可以称为代码补全接口，Python PSI也可以称为代码解析器，Python代码表示构造器也可以称为代码表示构造器，Python模型推理服务也可以称为推理装置。

[0149] 本实施例为了实现在PyCharm中python代码token补全功能，训练阶段具体实施流程如7B所示，具体实施步骤如下：

[0150] (一)从GitHub上获取star数量大于等于1000的python语言相关的项目作为代码库。

[0151] (二)通过Python Psi将源代码解析为AST(PsiFile)：利用PyCharm提供的Python PSI中的PsiFileFactory.getInstance(project).createFileFromText这个方法解析得到代码库中以文件为单位的源代码的PsiFile。获取到PsiFile后，遍历PsiFile的所有child并保留所有的PyClass以及PyFunction。

[0152] (三)基于PsiFile通过程序分析构建代码关系图及文本信息：(1)对于PyClass，保留类名并按照空格对类名进行切分原始token(记为token1)；其次，按照标点符号对token1进行切分并记为token2；随后，按照驼峰命名法、snake命名法以及数字对token2进行切分记为token3；最后，通过bpe对token3进行分词。对于PyFunction的函数名以及参数名进行同样的分词操作。然后将分词后的所有原子token进行去重，从而组成文本表示。(2)对于PyFunction，通过继承PyElementVisitor来遍历PyFunction中的每条语句。通过重写PyElementVisitor中不同类型语句的visit函数来实现不同类型语句的解析。例如，通过visitPyCallExpression来解析python中的函数调用语句。按序解析每条语句时，每条语句会与其前后语句所表示的结点之间连接一条控制流的边。每条语句对应的结点会存储这条语句的类型作为type标签以及这条语句分词之后的token序列作为token标签。如果token序列的长度超过了阈值(例如阈值为20)，那么该结点会被拆分为多个(即token序列长度除以阈值)孩子结点并且按序连接，结点两两之间是追加控制流的边。如果正在解析的语句为声明或赋值语句，那么我们会记录该变量并且会判断该变量在哪些语句中使用到，那么使用到该变量的语句与该变量所在结点之间就会产生数据流的边，如果已存在控制流的边，那么边的类型改为控制流和数据流。对于控制结点，我们会产生结构化占位符。例如，对于while语句，会先产生一个while结点，以及Condition结点、Body结点以及OutControl结点作为while结点的三个孩子结点分别表征while结点的不同分支，随后再分别解析每个分支中的代码。当所有语句解析完成后，我们可以得到一个全联通的代码关系图。

[0153] (四)基于得到的代码关系图，构造训练数据。具体可以参考上文对图2中步骤206的介绍实现。

[0154] (五)基于构造得到的训练数据，利用Tensorflow设计代码补全模型并进行训练。其中RNN系列Layer我们选择GRU，GNN选择Gated Graph Neural Network，聚合层选择sum操作。模型的learning rate设置为0.005，优化器设置为Momentum。Encoder中的embedding size设置为300，hidden size设置为300，GRU层数设置为3。Decoder中的embedding size设置为300，hidden size设置为600，GRU层数设置为3。

[0155] 本实施例为了实现在PyCharm中python代码token补全功能，推理阶段具体实施流程如图7C所示，具体实施步骤如下：

[0156] (一)继承com.intellij.codelnsight.completion.CompletionContributor类并重写fillCompletionVariants函数并在plugin.xml的extensions中进行注册,从而作为python代码token补全的入口。(对应图7A中的代码补全接口)。

[0157] (二)捕获光标所在函数:首先通过editor.getCaretModel().getOffset()获取光标所在位置的偏移量offset;其次通过psiFile.getElementAt函数获取偏移量所对应的element;最后通过PsiTreeUtil.getParentOfType函数获取光标坐在函数PyFunction。(对应图7A中的代码解析器)。

[0158] (三)通过Python PSI将光标所在函数解析为AST:将PyFunction展开得到其AST,从而获取到PyFunction所在的PyClass。(对应图7A中的代码解析器)。

[0159] (四)基于AST通过程序分析构建代码关系图以及文本信息:基于PyClass以及PyFunction,通过训练阶段步骤(三)中的(1)(2)步骤得到代码关系图以及文本信息。随后在代码关系图中光标所在位置添加对应的窟窿结点。(对应图7A中的代码表示构造器)。

[0160] (五)调用代码补全模型并结合束搜索(beam search)算法进行推理,输出补全结果:基于步骤(四)中得到的代码关系图以及文本信息为输入,输入到训练好的中小模型中并且利用束搜索算法通过概率高低(保留概率高的,过滤概率低的)来为推理结果进行排序。最终,将推理结果装载进入下拉选择框中展示给用户。(对应图7A中的推理装置)。

[0161] 通过实施例1的方案,可以提高待补全的Python代码的预测准确率。

[0162] 接下来,在实施例2中,以基于TypeScript语言编写的代码为例,举例介绍本申请实施例提供的方案。具体如下,

[0163] 当用户在的代码编辑器的function(即所写好的TypeScript function)内部键入字符后,触发TypeScript代码token补全动作,利用用户本地机器的计算资源,调用模型进行推理服务,最终返回结果给用户。

[0164] 实施例2的组件结构也可以如图7A所示。本实施例涉及到的模块包括代码token补全入口、TypeScript PSI、TypeScript代码表示构造器和TypeScript模型推理服务。其中,代码token补全入口也可以称为代码补全接口,TypeScript PSI可以称为代码解析器,TypeScript代码表示构造器也可以称为代码表示构造器,TypeScript模型推理服务也可以称为推理装置。

[0165] 本实施例为了实现TypeScript代码token补全功能,训练阶段具体实施流程也可参考图7B所示方案,其具体实施步骤与实施例1中训练阶段的具体实施步骤基本相同,区别在于:

[0166] (一)从GitHub上获取star数量大于等于1000的TypeScript语言相关的项目作为代码库。

[0167] (二)使用TypeScript PSI。PyClass被替换为TypeScript对应的Class以及PyFunction被替换为TypeScript对应的Function。

[0168] (三)继承JavaScriptVisitor进行代码关系图以及文本表示的解析。

[0169] 本实施例为了实现TypeScript代码token补全功能,推理阶段具体实施流程也可以参考图7C所示方案,其具体实施步骤与实施例一中训练阶段的具体实施步骤基本相同,区别在于:

[0170] (一)使用TypeScript PSI。PyClass被替换为ypeScript对应的Class以及

PyFunction被替换为TypeScript对应的Function。

[0171] (二)使用TypeScript模型进行推理服务。

[0172] 通过实施例2的方案,可以提高待补全的TypeScript代码的预测准确率。

[0173] 基于上文所描述的代码补全方案,本申请实施例提供了一种利用神经网络补全代码的方法,其中,该神经网络包括编码子网络和解码子网络。如图8所示,该方法包括如下步骤。

[0174] 步骤801,获取待补全代码的代码上下文。具体可以参考上文对图5中步骤501的介绍实现,在此不再赘述。

[0175] 步骤802,构建代码关系图,所述代码关系图包括对应所述代码上下文的至少一个结点、对应所述待补全代码的第一结点、至少一个连接边,其中,所述至少一个连接边包括所述第一结点和所述至少一个结点中的第二结点之间的第一连接边。具体可以参考上文对图5中步骤504的介绍实现,在此不再赘述。

[0176] 步骤803,在所述编码子网络,基于所述至少一个结点和所述第一结点的标签、所述至少一个连接边的标签,得到所述代码关系图的图特征向量,其中,所述至少一个结点的标签包括所述代码上下文的代码序列,所述第一结点和所述第一连接边的标签是预设的。具体可以参考上文对图5中506的介绍实现,在此不再赘述。

[0177] 步骤804,在所述解码子网络,至少基于所述图特征向量,得到所述待补全代码的代码序列。具体可以参考上文对图5中507的介绍实现,在此不再赘述。

[0178] 在一些实施例中,所述至少一个连接边还包括第二连接边,所述第二连接边连接所述至少一个结点中的第三结点和第四结点,所述第二连接边的标签为所述第三结点和所述第四结点之间的流关系,所述流关系为控制流和/或数据流。具体可以参考上文对图5中步骤504的介绍实现,在此不再赘述。

[0179] 在一些实施例中,所述基于所述至少一个结点和所述第一结点的标签、所述至少一个连接边的标签,得到所述代码关系图的图特征向量,包括:基于所述至少一个结点和所述第一结点的标签、所述至少一个连接边的标签,得到所述至少一个结点和所述第一结点的特征向量、以及所述至少一个连接边的特征向量;将所述至少一个结点和所述第一结点的特征向量、所述第一结点的特征向量、以及所述至少一个连接边的特征向量,融合为所述图特征向量。具体可以参考上文对图5中506的介绍实现,在此不再赘述。

[0180] 在该实施例的一个示例中,所述基于所述至少一个结点和所述第一结点的标签、所述至少一个连接边的标签,得到所述至少一个结点和所述第一结点的特征向量、以及所述至少一个连接边的特征向量,包括:基于所述第一结点的标签,得到所述第一结点的初始特征向量;基于所述第二结点的标签,得到所述第二结点的初始特征向量;以及基于所述第一连接边的标签,得到所述第一连接边的初始特征向量;基于所述第一结点的初始特征向量和所述第一连接边的初始特征向量,得到所述第一结点的特征向量;以及基于所述第二结点的初始特征向量和所述第一连接边的初始特征向量,得到所述第二结点的特征向量;基于所述第一结点的特征向量、所述第二结点特征向量和所述第一连接边的初始特征向量,得到所述第一连接边的特征向量。具体可以参考上文对图5中506的介绍实现,在此不再赘述。

[0181] 在该示例的一个例子中,所述编码子网络包括第一层和第二层;其中,所述第一结

点的初始特征向量和所述第二结点的初始特征向量是在所述第一层得到的，所述第一连接边的初始特征向量、所述第一结点的特征向量、所述第二结点的特征向量和所述第一连接边的特征向量是在所述第二层得到的。具体可以参考上文对图4所示代码补全模型的介绍实现，在此不再赘述。

[0182] 在一些实施例中，所述至少一个结点的标签还包括所述代码上下文在抽象语法树中的类型。具体可以参考上文对图5中步骤504的介绍实现，在此不再赘述。

[0183] 在一些实施例中，所述编码子网络还包括第三层；所述方法还包括：在所述第三层，基于所述代码上下文的文本信息，得到代码上下文的文本语义特征向量；所述文本信息包括所述代码上下文中的方法名、参数名、变量名、类名中的至少一项；所述至少基于所述图特征向量，得到所述待补全代码的代码序列包括：基于所述图特征向量和所述文本语义特征向量融合后的特征向量，得到所述待补全代码的代码序列。

[0184] 在一些实施例中，所述第二结点对应的代码的位置与所述待补全代码的位置相邻。

[0185] 在一些实施例中，所述代码序列为代码的标识符序列。

[0186] 在一些实施例中，所述神经网络的参数量小于1亿。具体可以参考上文对图4所示代码补全模型的介绍实现。

[0187] 本申请实施例提供的代码补全方法，可以将代码上下文的代码序列，以及代码上下文和代码补全代码的之间的结构信息，融合到代码关系图中。在神经网络中，进一步融合代码序列和代码间的结构信息，得到代码关系图的图特征，从而实现了从多个角度刻画代码上下文和待补全代码之间的关联语义。根据代码关系图的图特征，预测待补全代码的代码序列，可以提升待补全代码的预测准确率。

[0188] 基于上文描述的代码补全模型的训练方案，本申请实施例提供了一种训练神经网络的方法，该神经网络包括编码子网络和解码子网络。如图9所示，该方法包括如下步骤。

[0189] 步骤901，获取待补全代码的代码上下文。具体可以参考上文对图2中步骤201的介绍实现。

[0190] 步骤902，构建代码关系图，所述代码关系图包括对应所述代码上下文的至少一个结点、对应所述待补全代码的第一结点、至少一个连接边，其中，所述至少一个连接边包括所述第一结点和所述至少一个结点中的第二结点之间的第一连接边。具体可以参考上文对图2中步骤204的介绍实现。

[0191] 步骤903，在所述编码子网络，基于所述至少一个结点和所述第一结点的标签、所述至少一个连接边的标签，得到所述代码关系图的图特征向量，其中，所述至少一个结点的标签包括所述代码上下文的代码序列，所述第一结点和所述第一连接边的标签是预设的。具体可以参考上文对图2中步骤208的介绍实现。

[0192] 步骤904，在所述解码子网络，至少基于所述图特征向量，得到所述待补全代码的预测代码序列。具体可以参考上文对图2中步骤209的介绍实现。

[0193] 步骤905，基于所述预测代码序列和所述待补全代码的真实代码序列，得到预测损失。具体可以参考上文对图2中步骤210的介绍实现。

[0194] 步骤906，朝着所述预测损失减少的方向，更新所述神经网络。具体可以参考上文对图2中步骤211的介绍实现。

[0195] 在一些实施例中，所述神经网络的参数量小于1亿。

[0196] 本申请实施例提供的训练方法训练的神经网络，可以将代码上下文的代码序列，以及代码上下文和代码补全代码的之间的结构信息，融合到代码关系图中。在神经网络中，进一步融合代码序列和代码间的结构信息，得到代码关系图的图特征，从而实现了从多个角度刻画代码上下文和待补全代码之间的关联语义。根据代码关系图的图特征，预测待补全代码的代码序列，可以提升待补全代码的预测准确率。

[0197] 本申请实施例提供了一种利用神经网络代码补全的装置1000。其中，该神经网络包括编码子网络和解码子网络。如图10所示，装置1000包括：

[0198] 获取单元1010，用于获取待补全代码的代码上下文；

[0199] 构建单元1020，用于构建代码关系图，所述代码关系图包括对应所述代码上下文的至少一个结点、对应所述待补全代码的第一结点、至少一个连接边，其中，所述至少一个连接边包括所述第一结点和所述至少一个结点中的第二结点之间的第一连接边；

[0200] 第一得到单元1030，用于在所述编码子网络，基于所述至少一个结点和所述第一结点的标签、所述至少一个连接边的标签，得到所述代码关系图的图特征向量，其中，所述至少一个结点的标签包括所述代码上下文的代码序列，所述第一结点和所述第一连接边的标签是预设的；

[0201] 第二得到单元1040，用于在所述解码子网络，至少基于所述图特征向量，得到所述待补全代码的代码序列。

[0202] 装置1000的各功能单元可以参考图5或图8所示的方法实施例实现，在此不再赘述。

[0203] 本申请实施例提供了一种神经网络训练装置1100。其中，该神经网络包括编码子网络和解码子网络。如图11所示，装置1100包括：

[0204] 获取单元1110，用于获取待补全代码的代码上下文；

[0205] 构建单元1120，用于构建代码关系图，所述代码关系图包括对应所述代码上下文的至少一个结点、对应所述待补全代码的第一结点、至少一个连接边，其中，所述至少一个连接边包括所述第一结点和所述至少一个结点中的第二结点之间的第一连接边；

[0206] 第一得到单元1130，用于在所述编码子网络，基于所述至少一个结点和所述第一结点的标签、所述至少一个连接边的标签，得到所述代码关系图的图特征向量，其中，所述至少一个结点的标签包括所述代码上下文的代码序列，所述第一结点和所述第一连接边的标签是预设的；

[0207] 第二得到单元1140，用于在所述解码子网络，至少基于所述图特征向量，得到所述待补全代码的预测代码序列；

[0208] 第三得到单元1150，用于基于所述预测代码序列和所述待补全代码的真实代码序列，得到预测损失；

[0209] 更新单元1160，用于朝着所述预测损失减少的方向，更新所述神经网络。

[0210] 装置1100的各功能单元可以参考图2所示的方法实施例实现，在此不再赘述。

[0211] 上文主要从方法流程的角度对本申请实施例提供的装置进行了介绍。可以理解的是，各个终端为了实现上述功能，其包含了执行各个功能相应的硬件结构和/或软件模块。本领域技术人员应该很容易意识到，结合本文中所公开的实施例描述的各示例的单元及算

法步骤，本申请能够以硬件或硬件和计算机软件的结合形式来实现。某个功能究竟以硬件还是计算机软件驱动硬件的方式来执行，取决于技术方案的特定应用和设计约束条件。专业技术人员可以对每个特定的应用来使用不同方法来实现所描述的功能，但是这种实现不应认为超出本申请的范围。

[0212] 参阅图12，本申请实施例提供了一种计算设备1200。计算设备1200可以包括处理器1210和存储器1220。存储器1220中存储有指令，该指令可被处理器1210执行。当该指令在被处理器1210执行时，计算设备1200可以执行图5或图8所示的各方法实施例。

[0213] 参阅图13，本申请实施例提供了一种计算设备1300。计算设备1300可以包括处理器1310和存储器1320。存储器1320中存储有指令，该指令可被处理器1310执行。当该指令在被处理器1310执行时，计算设备1300可以执行图2或图9所示的各方法实施例。

[0214] 本申请实施例还提供了一种包含指令的计算机程序产品。所述计算机程序产品可以是包含指令的，能够运行在计算设备上或被储存在任何可用介质中的软件或程序产品。当所述计算机程序产品在计算设备上运行时，使得计算设备执行图8所示方法。

[0215] 本申请实施例还提供了一种包含指令的计算机程序产品。所述计算机程序产品可以是包含指令的，能够运行在计算设备上或被储存在任何可用介质中的软件或程序产品。当所述计算机程序产品在计算设备上运行时，使得计算设备执行图9所示方法。

[0216] 本申请实施例还提供了一种计算机可读存储介质。所述计算机可读存储介质可以是计算设备能够存储的任何可用介质或者是包含一个或多个可用介质的数据中心等数据存储设备。所述可用介质可以是磁性介质，(例如，软盘、硬盘、磁带)、光介质(例如，DVD)、或者半导体介质(例如固态硬盘)等。该计算机可读存储介质包括指令，所述指令指示计算设备执行图8所示方法。

[0217] 本申请实施例还提供了一种计算机可读存储介质。所述计算机可读存储介质可以是计算设备能够存储的任何可用介质或者是包含一个或多个可用介质的数据中心等数据存储设备。所述可用介质可以是磁性介质，(例如，软盘、硬盘、磁带)、光介质(例如，DVD)、或者半导体介质(例如固态硬盘)等。该计算机可读存储介质包括指令，所述指令指示计算设备执行图9所示方法。

[0218] 最后应说明的是：以上实施例仅用以说明本发明的技术方案，而非对其限制；尽管参照前述实施例对本发明进行了详细的说明，本领域的普通技术人员应当理解：其依然可以对前述各实施例所记载的技术方案进行修改，或者对其中部分技术特征进行等同替换；而这些修改或者替换，并不使相应技术方案的本质脱离本发明各实施例技术方案的保护范围。

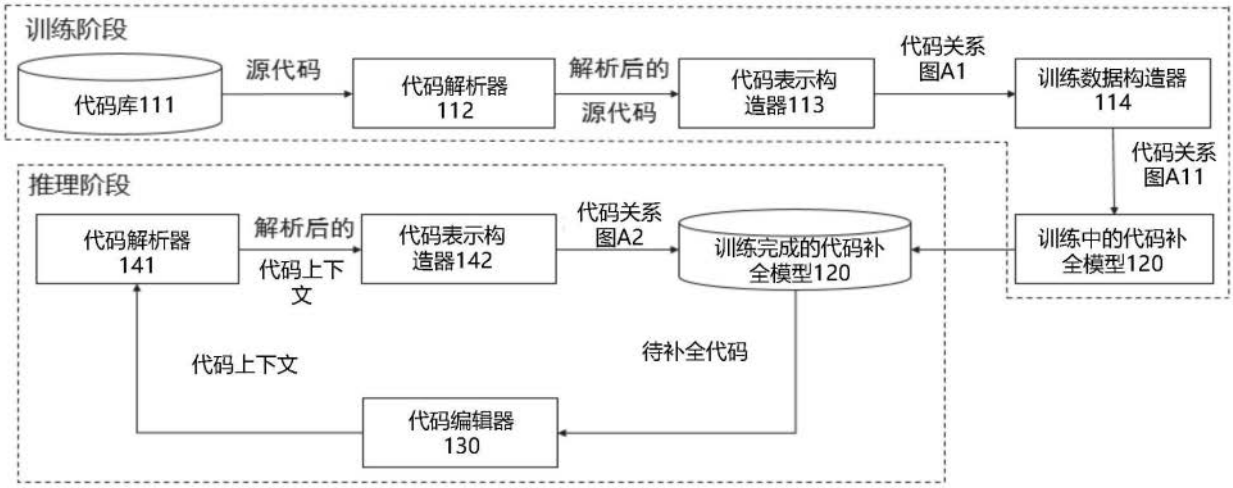

训练阶段
代码库111
源代码
代码解析器
112
解析后的
源代码
代码表示构
造器113
代码关系
图A1
训练数据构造器
114
代码关系
图A11
推理阶段
代码解析器
141
解析后的
代码上下
文
代码表示构
造器142
代码关系
图A2
训练完成的代码补
全模型120
训练中的代码补
全模型120
代码上下文
待补全代码
代码编辑器
130

图1

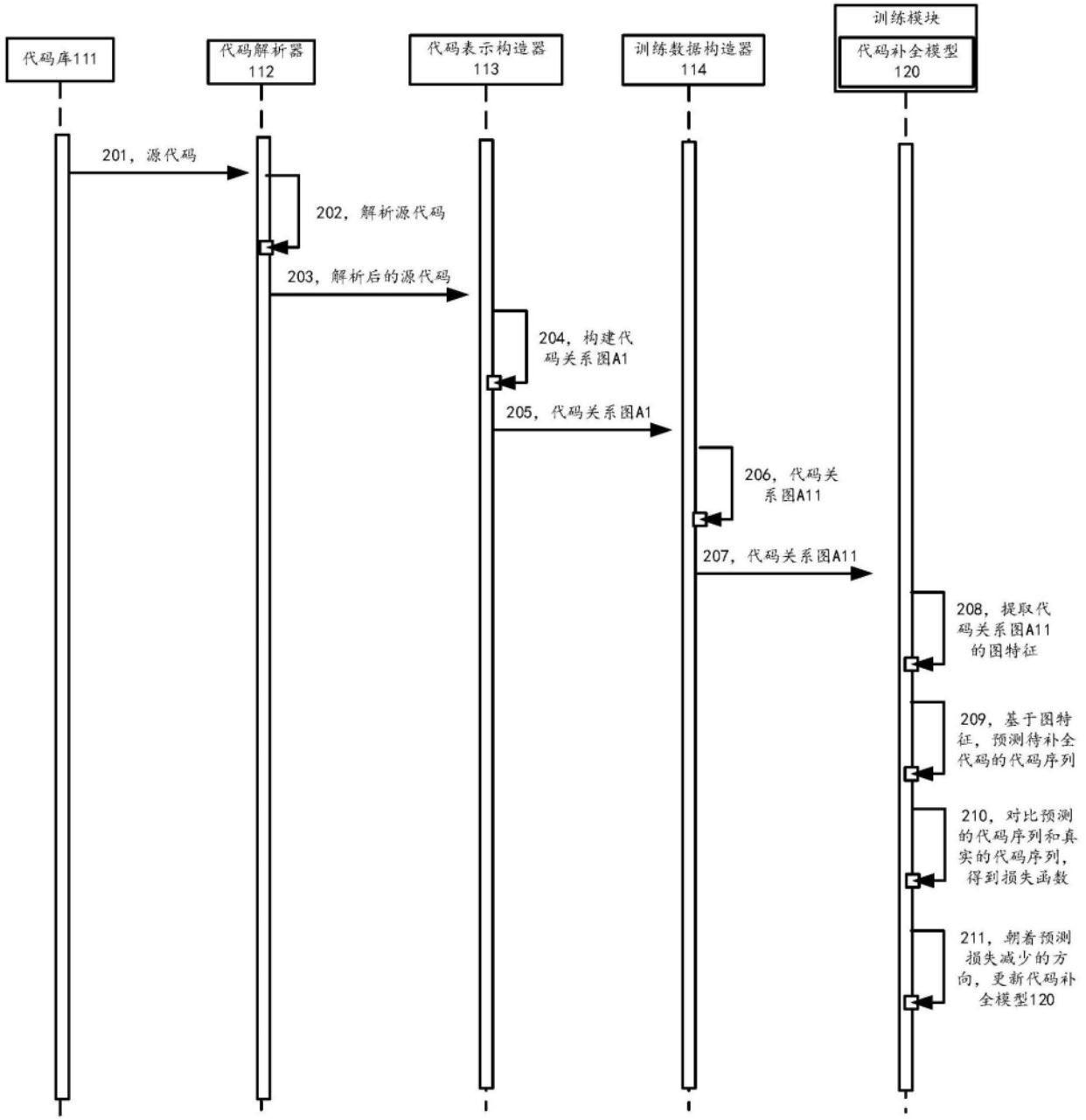

代码库111
代码解析器
112
代码表示构造器
113
训练数据构造器
114
训练模块
代码补全模型
120
201，源代码
202，解析源代码
203，解析后的源代码
204，构建代码关系图A1
205，代码关系图A1
206，代码关系图A11
207，代码关系图A11
208，提取代码关系图A11的图特征
209，基于图特征，预测待补全代码的代码序列
210，对比预测的代码序列和真实的代码序列，得到损失函数
211，朝着预测损失减少的方向，更新代码补全模型120

图2

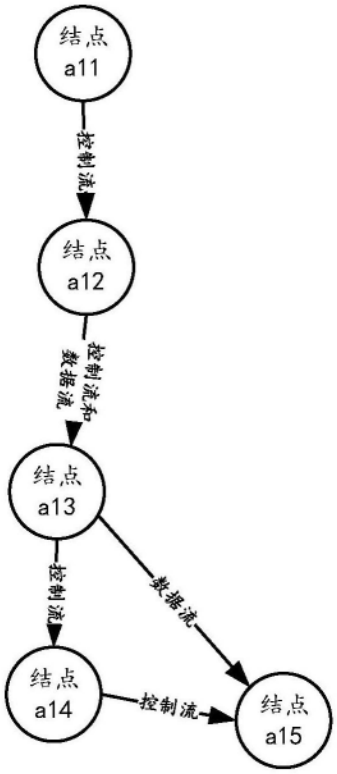

结点
a11
控制流
结点
a12
控制流和
数据流
结点
a13
控制流
数据流
结点
a14
控制流
结点
a15

图3A

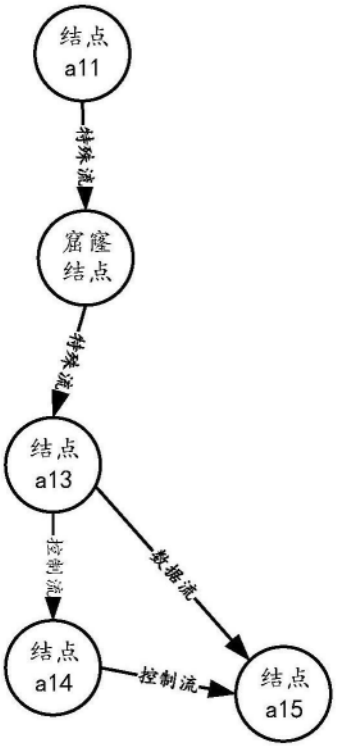

结点
a11
特殊流
窟窿
结点
特殊流
结点
a13
控制流
数据流
结点
a14
控制流
结点
a15

图3B

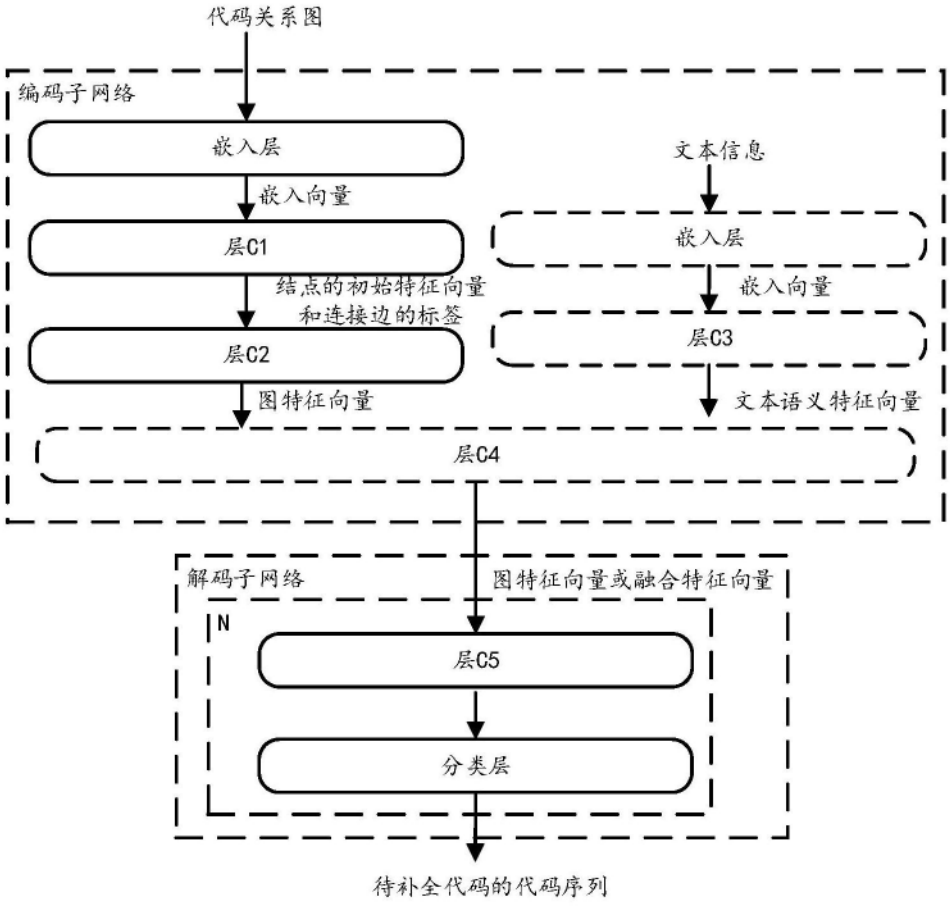

代码关系图
编码子网络
嵌入层
嵌入向量
层C1
结点的初始特征向量
和连接边的标签
层C2
图特征向量
文本信息
嵌入层
嵌入向量
层C3
文本语义特征向量
层C4
解码子网络
图特征向量或融合特征向量
N
层C5
分类层
待补全代码的代码序列

图4

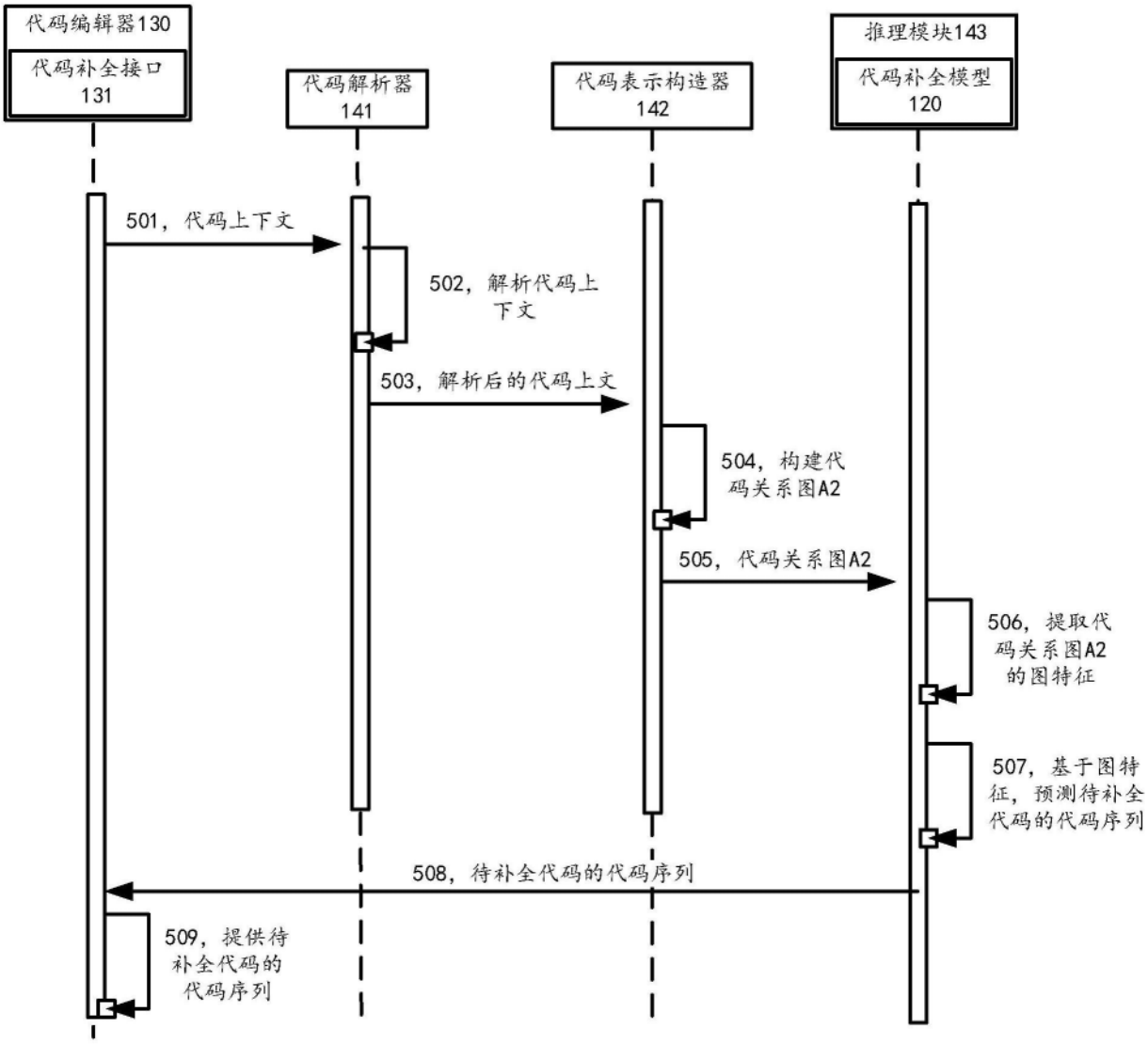

代码编辑器130
代码补全接口
131
代码解析器
141
代码表示构造器
142
推理模块143
代码补全模型
120
501，代码上下文
502，解析代码上
下文
503，解析后的代码上文
504，构建代
码关系图A2
505，代码关系图A2
506，提取代
码关系图A2
的图特征
507，基于图特
征，预测待补全
代码的代码序列
508，待补全代码的代码序列
509，提供待
补全代码的
代码序列

图5

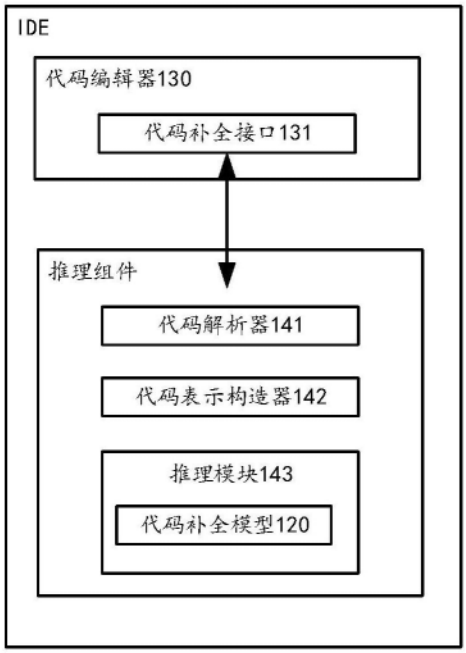

IDE
代码编辑器130
代码补全接口131
推理组件
代码解析器141
代码表示构造器142
推理模块143
代码补全模型120

图6

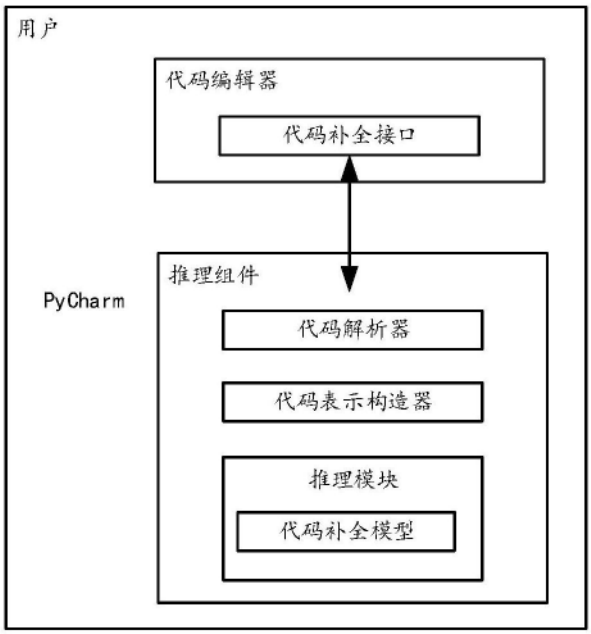

用户
代码编辑器
代码补全接口
PyCharm
推理组件
代码解析器
代码表示构造器
推理模块
代码补全模型

图7A

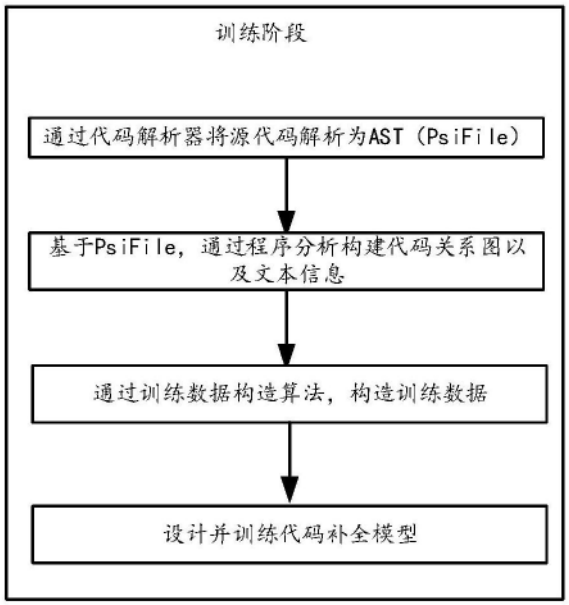

训练阶段
通过代码解析器将源代码解析为AST（PsiFile）
基于PsiFile，通过程序分析构建代码关系图以及文本信息
通过训练数据构造算法，构造训练数据
设计并训练代码补全模型

图7B

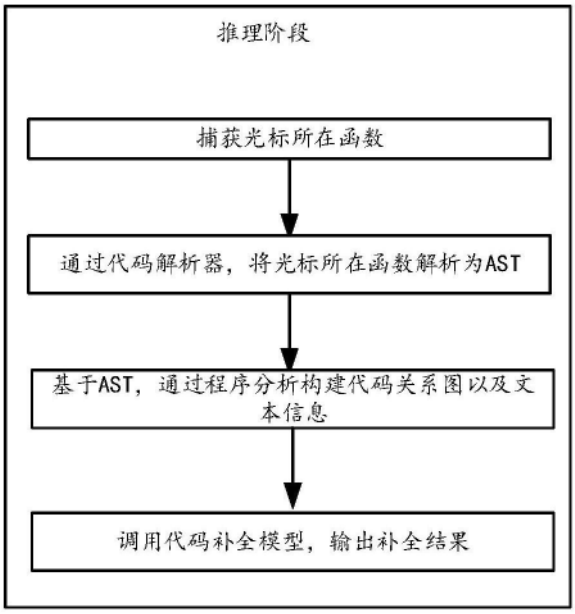

推理阶段
捕获光标所在函数
通过代码解析器，将光标所在函数解析为AST
基于AST，通过程序分析构建代码关系图以及文本信息
调用代码补全模型，输出补全结果

图7C

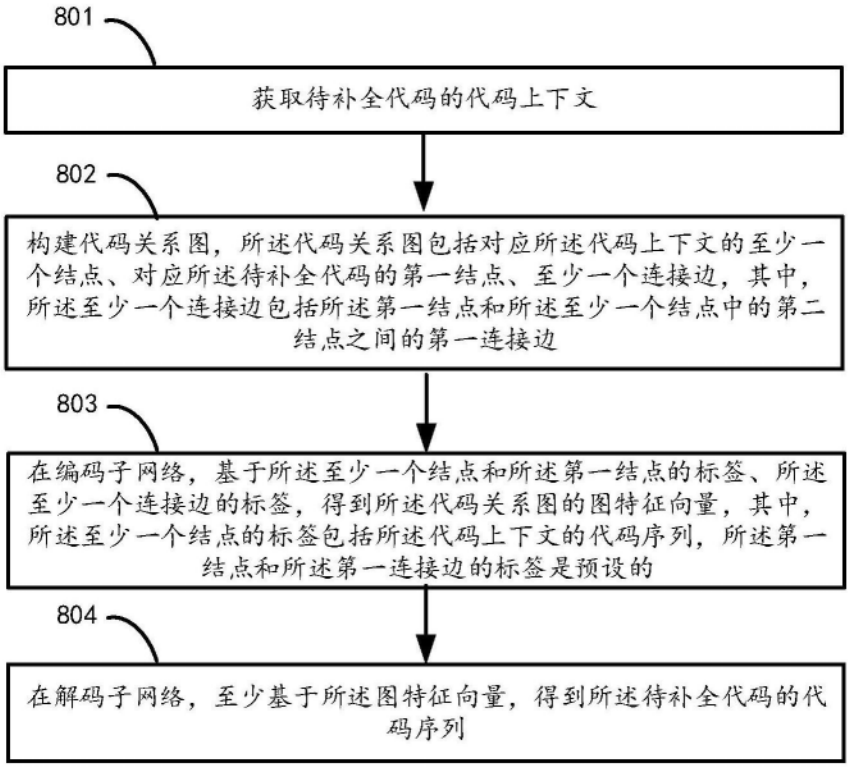

801
获取待补全代码的代码上下文
802
构建代码关系图，所述代码关系图包括对应所述代码上下文的至少一个结点、对应所述待补全代码的第一结点、至少一个连接边，其中，所述至少一个连接边包括所述第一结点和所述至少一个结点中的第二结点之间的第一连接边
803
在编码子网络，基于所述至少一个结点和所述第一结点的标签、所述至少一个连接边的标签，得到所述代码关系图的图特征向量，其中，所述至少一个结点的标签包括所述代码上下文的代码序列，所述第一结点和所述第一连接边的标签是预设的
804
在解码子网络，至少基于所述图特征向量，得到所述待补全代码的代码序列

图8

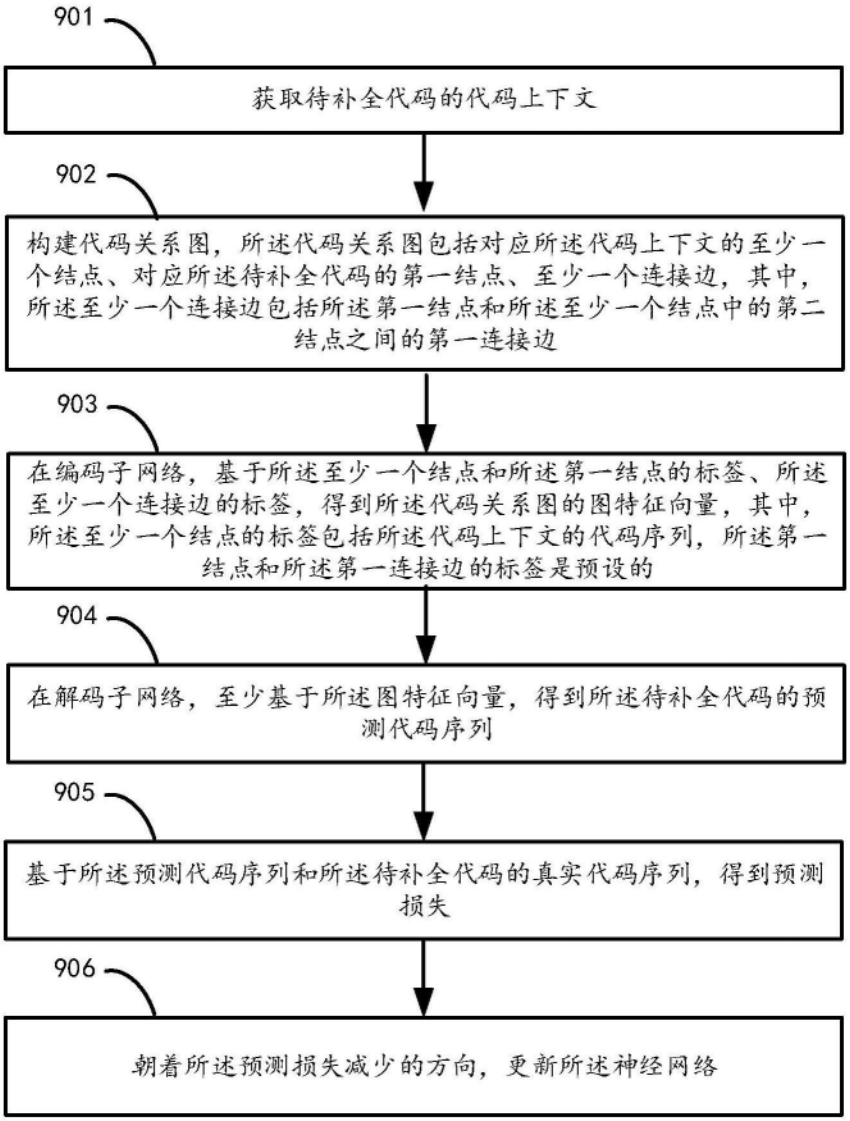

901
获取待补全代码的代码上下文
902
构建代码关系图，所述代码关系图包括对应所述代码上下文的至少一个结点、对应所述待补全代码的第一结点、至少一个连接边，其中，所述至少一个连接边包括所述第一结点和所述至少一个结点中的第二结点之间的第一连接边
903
在编码子网络，基于所述至少一个结点和所述第一结点的标签、所述至少一个连接边的标签，得到所述代码关系图的图特征向量，其中，所述至少一个结点的标签包括所述代码上下文的代码序列，所述第一结点和所述第一连接边的标签是预设的
904
在解码子网络，至少基于所述图特征向量，得到所述待补全代码的预测代码序列
905
基于所述预测代码序列和所述待补全代码的真实代码序列，得到预测损失
906
朝着所述预测损失减少的方向，更新所述神经网络

图9

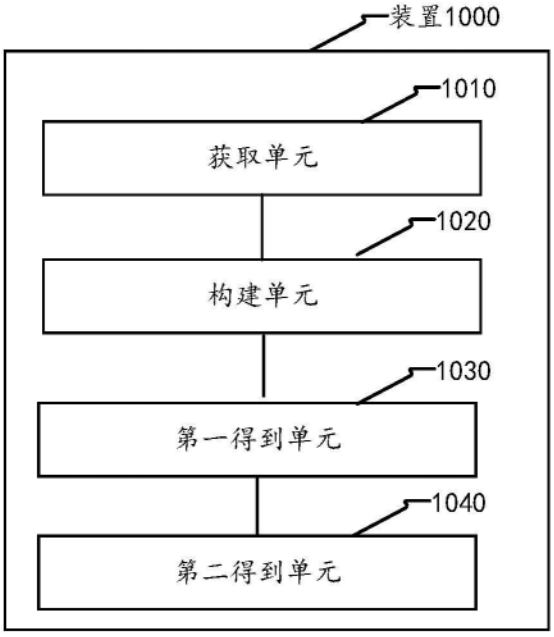

装置1000
1010
获取单元
1020
构建单元
1030
第一得到单元
1040
第二得到单元

图10

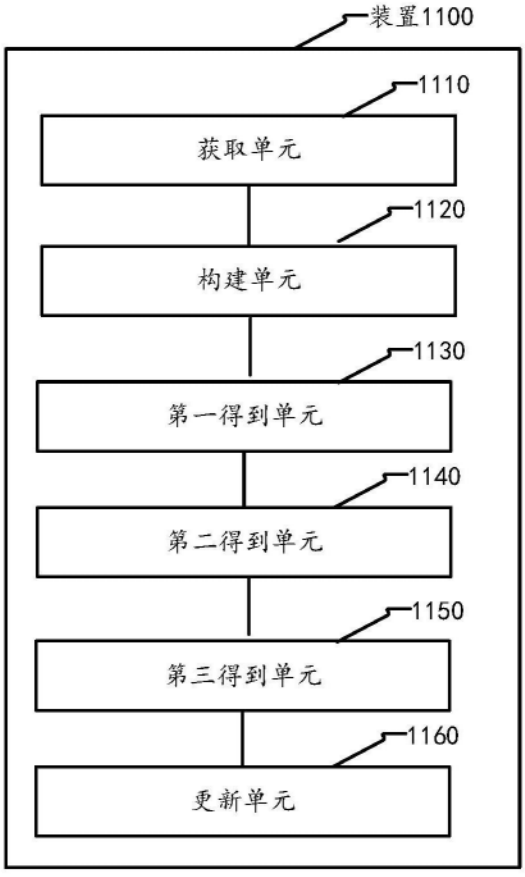

装置1100
1110
获取单元
1120
构建单元
1130
第一得到单元
1140
第二得到单元
1150
第三得到单元
1160
更新单元

图11

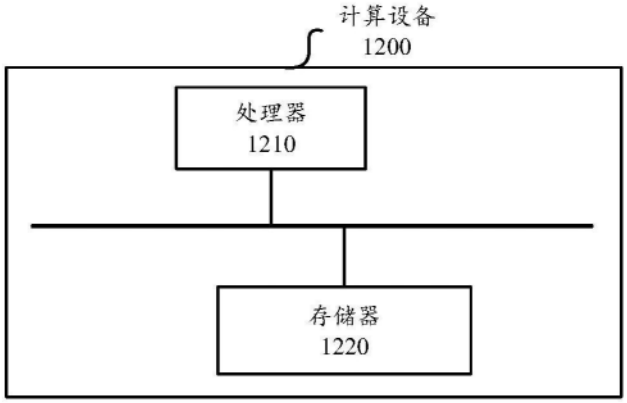


图12

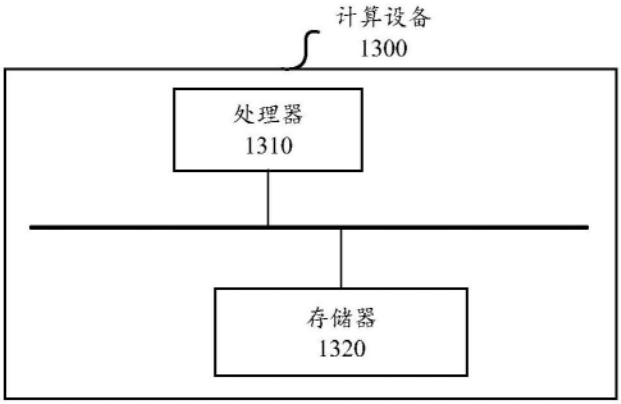


图13