title: 混合语义代码表示-论文笔记
date: 2021-07-21 15:22:09
categories:

- 论文笔记
  tags:
- 漏洞挖掘
- 代码表示学习

A hybrid code representation learning approach for predicting method names

来源信息

- The Journal of Systems & Software(**SCI 2区**)
- 机构：复旦大学
- 作者：Fengyi Zhang, Bihuan Chen, Rongfan Li, Xin Peng

Abstract

程序语义属性，如类名、方法名、变量名和类型，在软件开发和维护中起着重要作用。方法名称尤其重要。因为它们为开发人员提供了相互交流（如代码审查和程序理解）的抽象基础。现有的方法名预测方法通常将代码表示为词汇标记或语法AST（抽象语法树）路径，这使得它们很难学习代码语义，并阻碍了方法名预测的有效性。之前的工作已经尝试将代码表示为执行过程来捕获代码语义，但在收集执行过程时存在可伸缩性。
在本文中，作者提出了一种混合代码表示学习方法，即Meth2Seq，可以将方法编码为分布式向量序列。Meth2Seq由三部分构成 ：程序依赖性图上的一系列路径，类型化的中间表示语句的序列，一个自然语言注释的句子，以可伸缩地捕获代码语义。将Meth2Seq方法学习到的向量序列输入到解码器模型中以预测方法名。作者在67个Java项目中对280.5K个方法的评估表明，Meth2Seq在f1-score中比两种最先进的代码表示学习方法好92.6%和36.6%，同时也比两种最先进的方法名称预测方法多出85.6%和178.1%。

Introduction

Representation learning：将原始数据嵌入到低维向量，以保留原始数据的属性，并可输入下游深度学习模型。它允许模型学习如何提取这些特征，以及如何在特定的任务中使用它们。例如word embedding， paragraph embedding。
编程理解任务之一是预测程序语义属性，如类名、方法名、变量名和变量类型。这项任务非常重要，因为这种语义属性或标签在软件开发和维护过程中是常见和重要的。一方面，开发人员必须为他们在程序中声明的每个类、方法、变量和参数选择有意义的语义名称。另一方面，开发人员在代码审查或维护期间经常依赖名称来理解程序或API的行为。现有的方法主要有以下几种：

- 基于AST的预测程序语义属性的方法：将代码表示为语法级别上的一系列AST节点或路径序列，然后利用机器学习方法进行学习预测。（问题：很难学习代码语义）
- 基于执行方式的预测程序语义属性的方法：从语义级的符号和具体执行过程中学习代码表示。在预测程序属性方面取得了良好的准确性。（问题：收集执行过程时可能会遇到问题）
- 基于Token的预测程序语义属性的方法：将代码表示为词汇级别上的Token序列，并应用N-grim模型或深度神经网络模型来预测程序属性。（问题：很难学习代码语义）

Contributions：

- 论文提出了一种混合代码表示学习方法：Meth2Seq，以嵌入基于三种代码表示的混合的方法，即程序依赖图上的一系列路径，捕获语义结构，基于上下文自由语法的类型IR语句序列，用来描述每种IR语句类型的运算语义和自然语言注释的句子。
- 在67个Java项目中，使用280.5K方法的数据集评估了方法名预测任务上的Meth2Seq，它显著优于最先进的代码表示学习方法同时方法名预测上也优于现有的方法。

方法

论文的思想是使用程序依赖图(PDG)中的代码语义结构、中间表示法(IR)中的隐式操作语义和自然语言注释中的语义。特别是，PDG中的控制流和数据流分别描述了语句的执行顺序以及语句之间的数据定义和用法。与AST相比，PDG和IR都提供了更高的代码抽象。

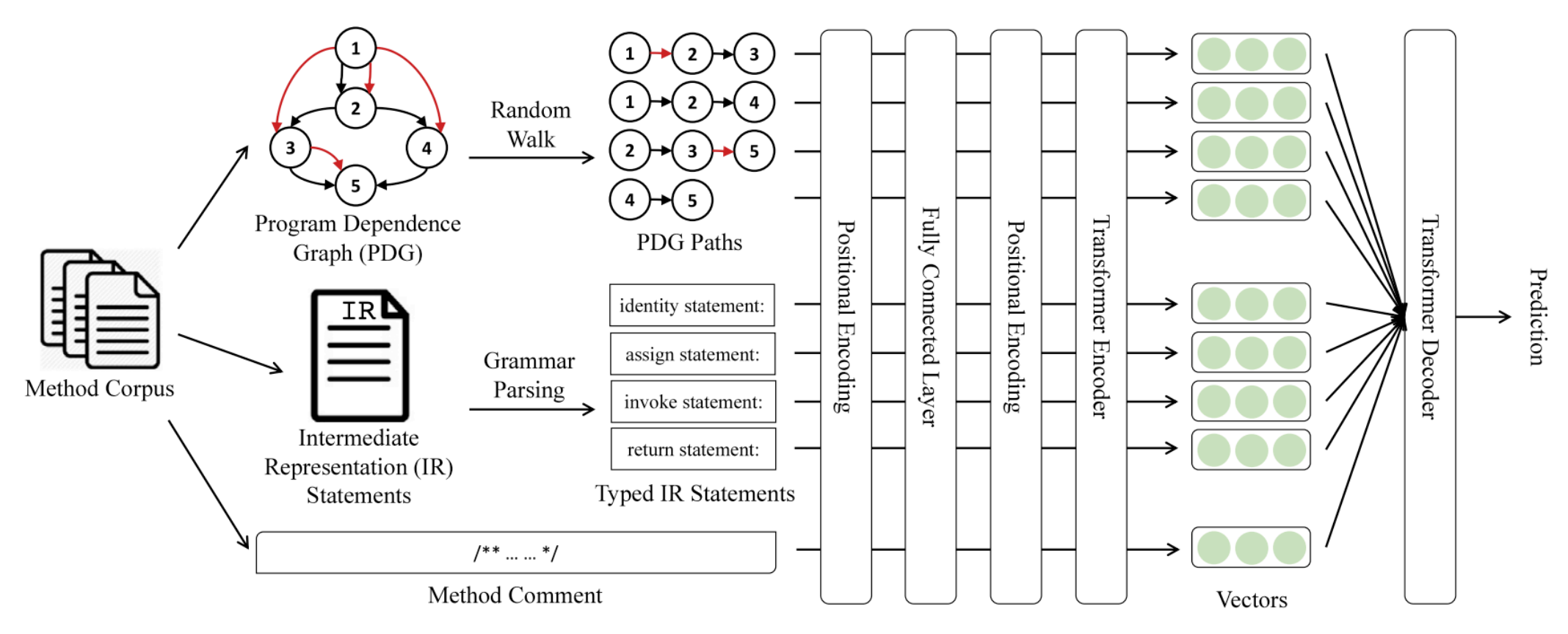

Fig. 2. Overall of our approach.

如图所示，它将方法主体表示为一系列PDG路径、类型化的IR语句序列和方法注释句。然后，它首先应用位置编码和一个全连接层来初始化每个PDG路径、每个类型的IR语句和方法注释作为一个向量。并使用位置编码和Transformer编码器为每个方法生成一个向量序列。学习到的向量序列可以输入给Transformer解码器来预测名称。

将方法表示为一系列的PDG路径

PDG：程序依赖图（由控制流图+数据流图构成）

```
protected String randomAvailableHostname() {
    List<InetAddress> available = info.nodeList();
    Collections.shuffle(available);
    return available.get(0).getHostName();
}
```

Fig. 3. An example of a method.

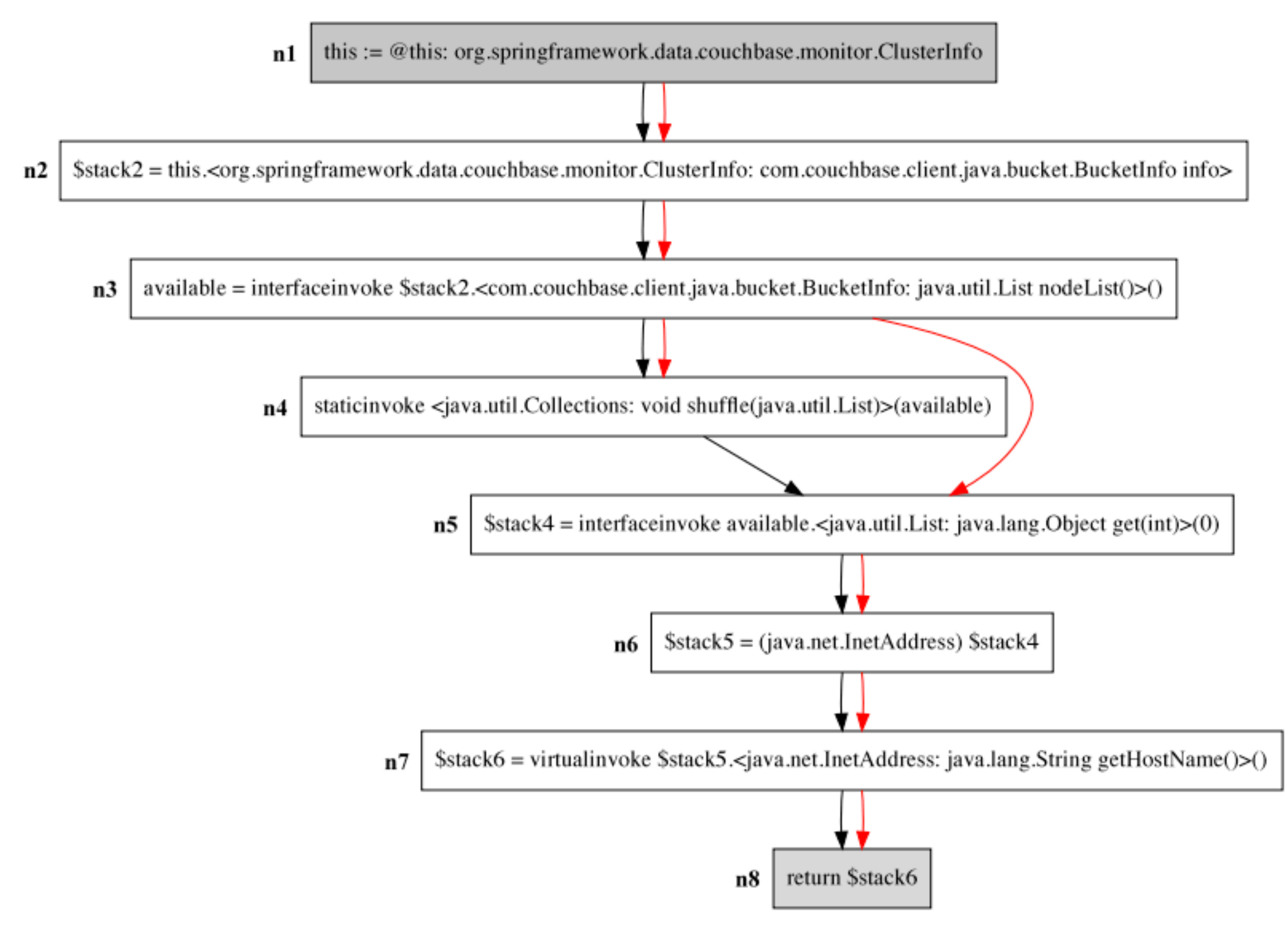

Fig. 4. The PDG of the method in Fig. 3.

如图所示，红色箭头表示数据依赖，黑色箭头表示控制依赖。

使用PDG路径表示代码的优势：

- PDG路径可以捕获不同的局部语义结构信息；
- PDG路径将非顺序图结构转换为序列；
- 可以自动生成PDG路径，并适用于其他编程语言；

PDG路径生成方法：随机游走策略，首先从N中均匀随机地选择一个节点作为当前节点n，然后进行迭代，直到达到最大长度k或达到出口节点ne，在每次迭代中，它均匀地从n的外边(n)中随机选择一条边缘，并使用该边缘的目标节点作为当前节点继续迭代，直到所有的节点都完成遍历。

K是如何确定的？每一个方法所生成的PDG的大小都是不一样的，生成的PDG数量最大为N，针对每一种方法，生成b* | N | 条路径，长度不超过|N|/a + 1。

将方法表示为类型化的IR语句序列

该过程分为两步：IR生成和类型IR表示。

- IR生成：使用Jimple作为IR，Jimple是Java字节码的一种类型化和紧凑的3-address代码表示。
- 类型IR表示：在Jimple中，共有15种类型的Jimple语句具有在其上下文自由语法中定义的操作语义。针对每一条语句，添加non-terminal表示该语句的类型的符号。然后，根据上下文自由语法解析每个Jimple语句，并为每个terminal符号添加它出现的no-terminal符号。基于这两条规则，可以将每个Jimple语句转换为比原始的Jimple语句更自然，可以减轻深度学习的负担。

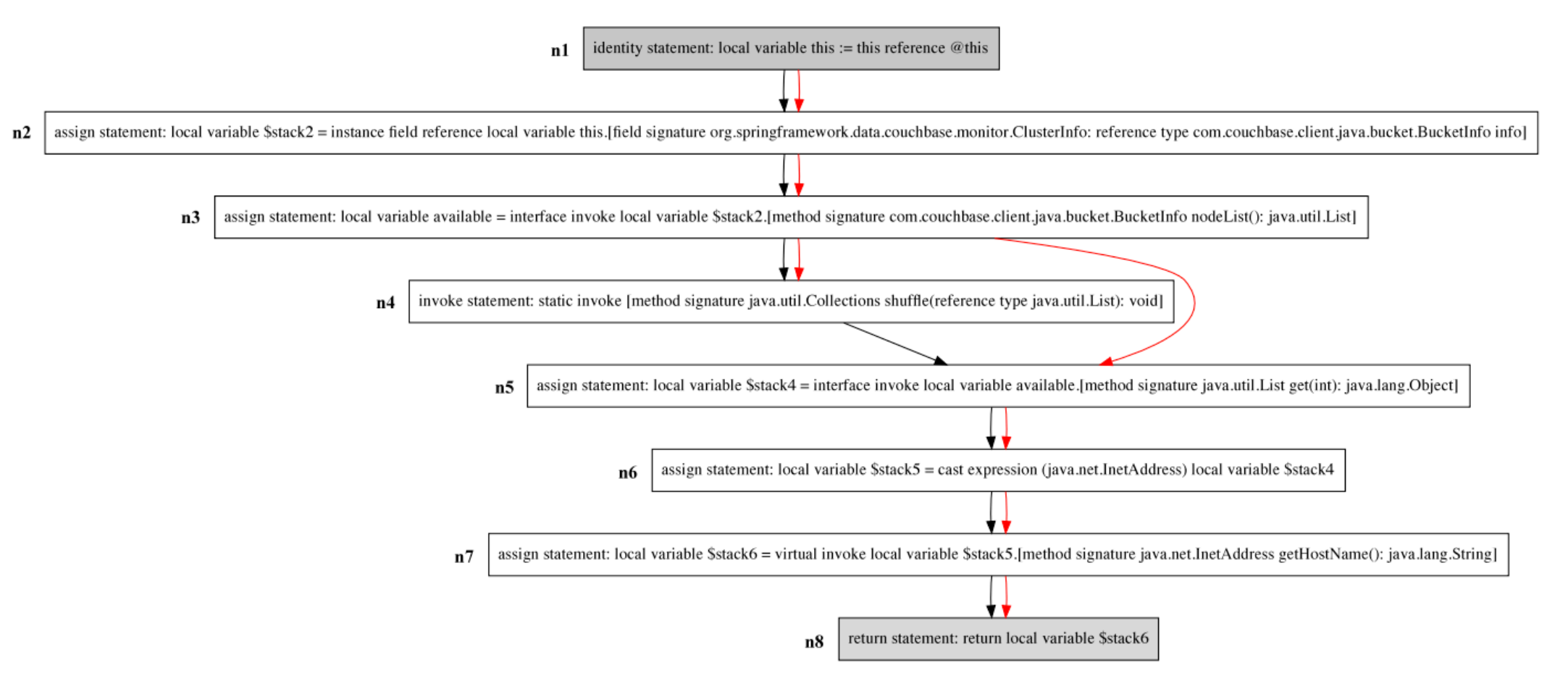

Fig. 5. Typed IR statements of the method in Fig. 3.

将方法表示为注释的句子

注释表示方法是简单地使用方法注释的第一句，因为第一句话通常是根据JavaDoc指导下的方法的总结。使用常用的策略（即tokenizing、删除停用词和符号）来预处理提取的句子。最后，对每一个方法生成一条注释，对于没有注释的方法，默认留空。

以一种混合的方式学习代码表示

将每一个方法定义为 |P| + |N | + 1，其中 | P | 为PDG路径， | N | 为IR语句，1表示一条注释。首先使用位置编码来编码每个PDG路径中的单词的位置信息、每个输入的IR语句和方法注释。然后，使用一个全连接层来将这个句子中的每个单词嵌入到一个向量中。然后，|P|+|N|+1向量按顺序输入位置编码，以编码位置信息(对于类型化IR语句的向量特别有用)，然后使用Transformer编码器，生成一系列以混合方式表示方法的|P|+|N|+1注意矢量。

预测方法名称

利用Meth2Seq生成的注意向量序列，使用Transformer解码器来实现方法名称预测的任务。为了构造输出词汇表，提取方法名并将它们作为输出词汇表。

Evaluation

Dataset

使用67个GitHub的Java项目。并创建了一个包含280.5K方法的语料库。7:1:2（训练：验证：测试）。

RQ1:与最先进的方法相比，Meth2Seq在预测方法名称方面的有效性如何?

Table 1
Quantitative comparative results in four metrics.

Approach	Exact	Precision	Recall	F1
CODE2VEC	0.296	0.407	0.399	0.403
CODE2SEQ	0.329	0.626	0.520	0.568
LIUNAME	0.227	0.457	0.385	0.418
HEMA	0.200	0.443	0.204	0.279
METH2SEQ	**0.582**	**0.778**	**0.778**	**0.776**

Table 2
Get/Set methods in all and exact match predictions.

Approach	All Prediction (%)	Exact Prediction (%)
CODE2VEC	30.9	44.5
CODE2SEQ	39.3	56.1
LIUNAME	35.9	39.6
HEMA	32.7	38.1
METH2SEQ	34.6	37.4

RQ2:Meth2Seq中的三种代码表示对所实现的有效性的贡献是什么?

Table 3
Code representation contribution analysis.

Approach	Exact	Precision	Recall	F1
METH2SEQ	**0.582**	**0.778**	**0.778**	**0.776**
METH2SEQ w/o PDG	0.475	0.571	0.585	0.573
METH2SEQ w/o IR	0.431	0.513	0.530	0.516
METH2SEQ w/o Comment	0.534	0.638	0.648	0.639
METH2SEQ w/o Type in IR	0.462	0.569	0.565	0.561
METH2SEQ with only PDG	0.414	0.472	0.509	0.489
METH2SEQ with only IR	0.460	0.548	0.541	0.544

RQ3:Meth2Seq对PDG路径数量和长度的敏感性是多少?

Table 4
Sensitivity analysis of the maximum path length.

Path Length $= \frac{|\mathcal{N}|}{a} + 1$	Exact	Precision	Recall	F1
a = 1	0.545	0.736	0.740	0.738
a = 2	0.562	0.755	0.753	0.754
a = 4	**0.582**	**0.778**	**0.778**	**0.776**
a = 8	0.569	0.762	0.759	0.760

Table 5
Sensitivity analysis of the number of paths.

Path Number $= b \times |\mathcal{N}$	Exact	Precision	Recall	F1
b = 2	0.538	0.734	0.732	0.733
b = 4	0.554	0.715	0.743	0.729
b = 6	0.565	0.746	0.753	0.750
b = 8	**0.582**	**0.778**	**0.778**	**0.776**