|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 论文题目 | Language-agnostic representation learning of source code from structure and context | | | |
| Paper URL | https://paperswithcode.com/paper/language-agnostic-representation-learning-of-1 | | | |
| Project URL | [GitHub - danielzuegner/code-transformer: Implementation of the paper "Language-agnostic representation learning of source code from structure and context".](https://github.com/danielzuegner/code-transformer) | | | |
| 综述/背景介绍 | 发展状况 | 原因 | 意义 | 关键词（速记词汇、信息索引词汇） |
| 代码的机器学习是一个活跃且不断发展的研究领域，其目标是构建能够学习丰富语义的程序表示的模型。源代码（上下文）及其解析的抽象语法树（AST；结构）是同一代码的两种互补表示。  传统上，机器学习模型的设计者主要依赖于结构或上下文。源代码表示自然适合来自自然语言处理(NLP)的模型，例如长期短期记忆网络。另一方面，利用结构表示的模型通常基于图神经网络(GNN)。虽然AST表示使源代码的高度结构化特性对模型是明确的，但由于大多数GNN使用消息传递框架，它们的学习表示存在局部性，难以利用远程交互。 Hellendorn et al.(2020)探索了可以利用几种表示法的模型，包括结构和上下文。他们的图关系嵌入注意力transformer(GREAT)  我们提出了一个新模型，它联合学习源代码的上下文和结构。与以前的方法相比，我们的模型仅使用与语言无感的特征，即可以直接从 AST 计算的源代码和特征。除了在这项工作中考虑的所有五种编程语言上获得最先进的单语言代码摘要外，我们还提出了第一个多语言代码摘要模型。结果表明，对来自多种编程语言的非并行数据进行联合训练可以提高所有单个语言的结果，其中最强的收益是在低资源语言上。值得注意的是，仅来自 Context 的多语言训练并没有带来相同的改进，突出了结合 Structure 和 Context 进行代码表示学习的好处。 | | | 上下文（context）  结构（structure）  联合学习（jointly learn）  与语言无感（language-agnostic）  程序表示（programs representation）  多语言代码摘要模型（ multilingual code summarization model） |
| 假设 | AST作为各编程语言的一种中间表示，故而从这种中间表示出发，将AST的节点和源代码的token对应到，采用各对节点的各种关系距离来构造一种新的代码编码时的输入。这样使输入信息的信息量丰富且有较强的广泛性。 | | |  |
| 方法描述(含图) | 我们提出了code TRANSFORMER，它在自注意操作中结合了基于结构和上下文计算的距离。为了结合上下文和结构信息，我们将序列中的每个token分配给一个AST节点，方法是选择源代码中包含token的范围最短的AST节点。我们将(子)token嵌入与token的指定AST节点类型以及token化返回的token类型的嵌入串联在一起作为注意力层的输入我们通过计算AST上的成对距离(如最短路径长度)，使模型在每一层都可以访问完整的结构。在注意力计算中使用相对距离而不是绝对位置。重要的是，我们的所有特性都是语言无感的，也就是说，可以根据源代码和AST很容易地计算出任何编程语言的特性。  我们还训练了第一个用于代码摘要的多语言模型。这是因为我们的模型只使用与语言无感的特性，这些特性可以很容易地在任何编程语言中获得。值得注意的是，在多种编程语言上训练我们的模型大大提高了所有语言的性能。此外，仅从上下文进行的多语言培训不会带来同样的改进，它突出了将结构和上下文结合起来对代码表示学习的好处。 | | |  |
| 实验设计 |  | | |  |
| 数据处理 | 输入 | 处理方式 | | 关键词（速记词汇、信息索引词汇） |
| 源代码和AST | 1、对源代码进行token化并删除注释  2、计算AST中的token和节点之间的映射。每个token都被分配到AST中源范围最短的节点上，而该节点仍然包含了token的源范围。  3、计算节点的距离（最短路径、祖先距离、同族距离）  4、使用token之间的不同距离(例如AST或个性化PageRank上的最短路径)来推断它们的相对位置  5、每个输入token都有一个token类型(来自Pygments token化器)和一个AST节点类型。AST节点类型是分配给每个相应token的节点类型，我们连接多个子token、token类型和AST节点类型的嵌入。然后，我们应用线性层(没有激活函数)向下投影到模型的嵌入维度 | |  |
| 结论 | 我们提出了code transformer，它从程序的结构和上下文中联合学习，而只依赖于语言无感的特征。我们的模型在五种不同的编程语言上获得了最好的的代码摘要性能。除了这些针对单个语言的训练结果外，我们模型的语言无感允许我们在多种编程语言上进行联合训练。由此产生的多语言模型在所有编程语言上的表现都大大优于它的单语言变体，为每种语言设定了最先进的技术水平。我们观察到，在资源最少的语言上，多语言训练的提升最大，这表明多语言训练可以改善较少使用的编程语言的学习。值得注意的是，仅从语境出发的多语言培训不会带来同样的改善，这突出了结构和上下文相结合的好处。 | | | |
| 局限性分析 |  | | | |

**（论文名）：**从结构和上下文的源代码语言无感表示学习

**（题目）：**《Language-agnostic representation learning of source code from structure and context》

**（论文URL）：**论文地址: https://paperswithcode.com/paper/language-agnostic-representation-learning-of-1

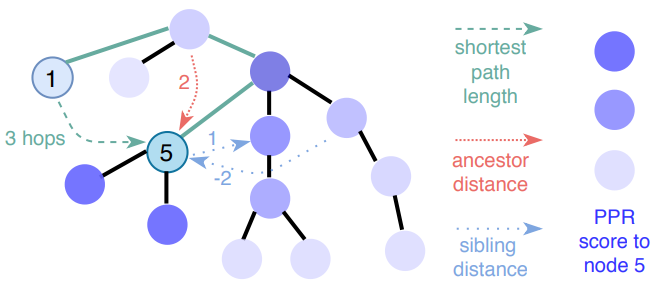
**（总结）：**我们提出了一个新模型：code transformer，它联合学习源代码的上下文和结构。传统上，机器学习模型的设计者主要依赖于结构或上下文。源代码表示自然适合来自自然语言处理(NLP)的模型，例如长期短期记忆网络。另一方面，利用结构表示的模型通常基于图神经网络(GNN)。与以前的方法相比，我们的模型仅使用与语言无感的特征，即可以直接从AST计算的源代码和特征。通过对transformer的自注意力进行改进，将绝对位置信息换成相对位置信息，而相对位置则通过token之间的不同距离(例如AST或个性化PageRank上的最短路径)来推断。为了结合上下文和结构信息，我们将序列中的每个token分配给一个AST节点，方法是选择源代码中包含token的范围最短的AST节点。我们将(子)token嵌入与token的指定AST节点类型以及token化返回的token类型的嵌入串联在一起作为注意力层的输入。除了在这项工作中考虑的所有五种编程语言上获得最先进的单语言代码摘要外，我们还提出了第一个多语言代码摘要模型。结果表明，对来自多种编程语言的非并行数据进行联合训练可以提高所有单个语言的结果，其中最强的收益是在低资源语言上。值得注意的是，仅来自 Context 的多语言训练并没有带来相同的改进，突出了结合 Structure 和 Context 进行代码表示学习的好处。

**（附图）：**

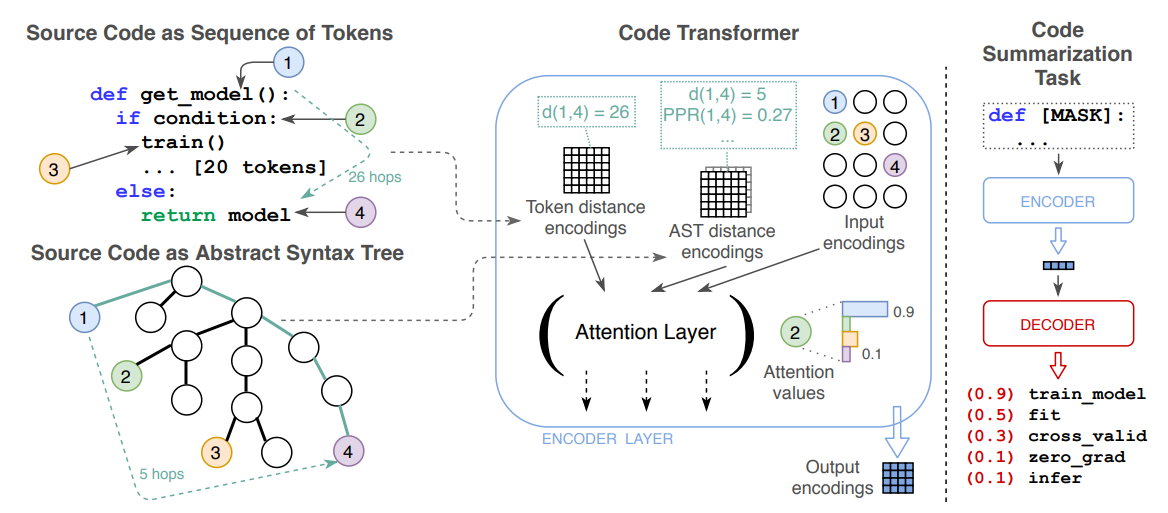
**自注意力公式改进，采用相对位置替换绝对位置信息**

****

**代码AST结构中节点对的各种距离表示和PPR，通过各种距离得到节点对的相对距离**



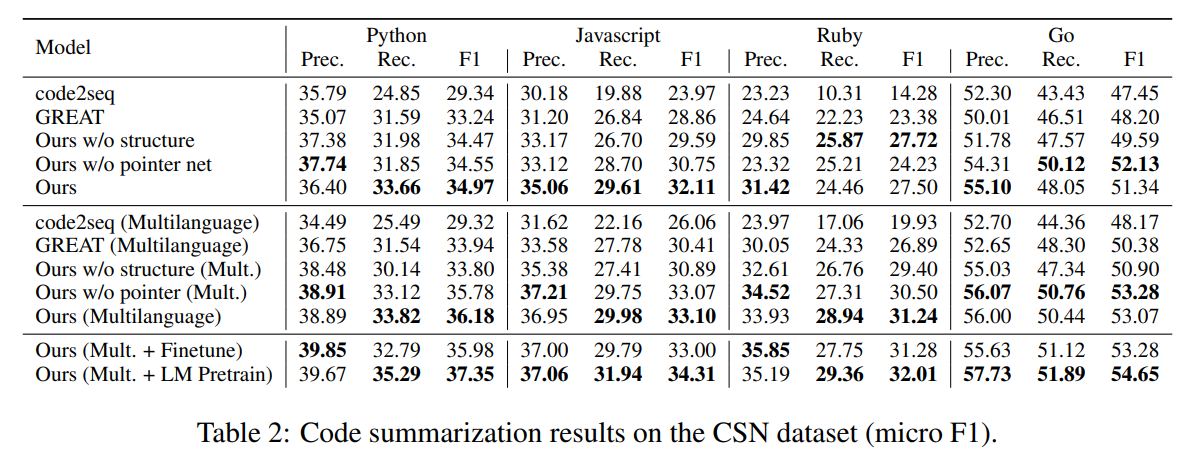
整体模型图



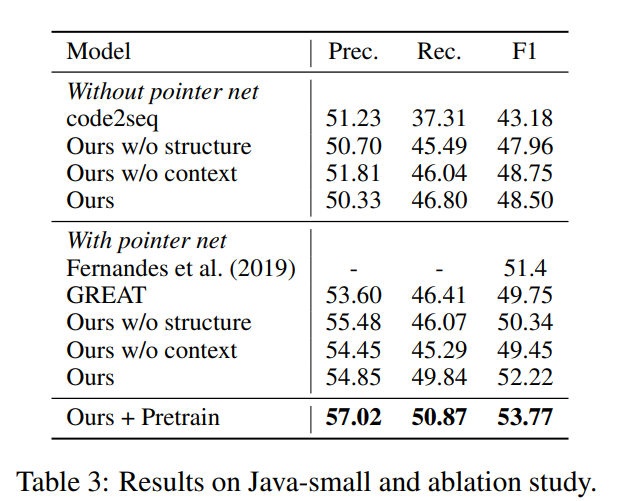
左：输入代码片段的序列(上下文)和AST(结构)表示。中心：代码transformer联合利用token序列和抽象语法树来学习源代码的表示形式。除了输入token和节点嵌入之外，该模型还使用token之间的不同距离(例如AST或个性化PageRank上的最短路径)来推断它们的相对位置。输出嵌入可用于下游任务，如代码汇总(右)。

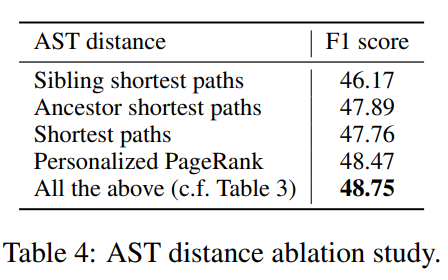
为了结合上下文和结构信息，我们将序列中的每个token分配给一个AST节点，方法是选择源代码中包含token的范围最短的AST节点。我们将(子)token嵌入与token的指定AST节点类型以及token化返回的token类型的嵌入串联在一起。在所有内部节点中，我们只使用与序列中的token相对应的节点作为输入；但是，剩余的内部节点可以由模型使用，因为它们的存在会影响其余AST节点之间的距离。

实验结果：



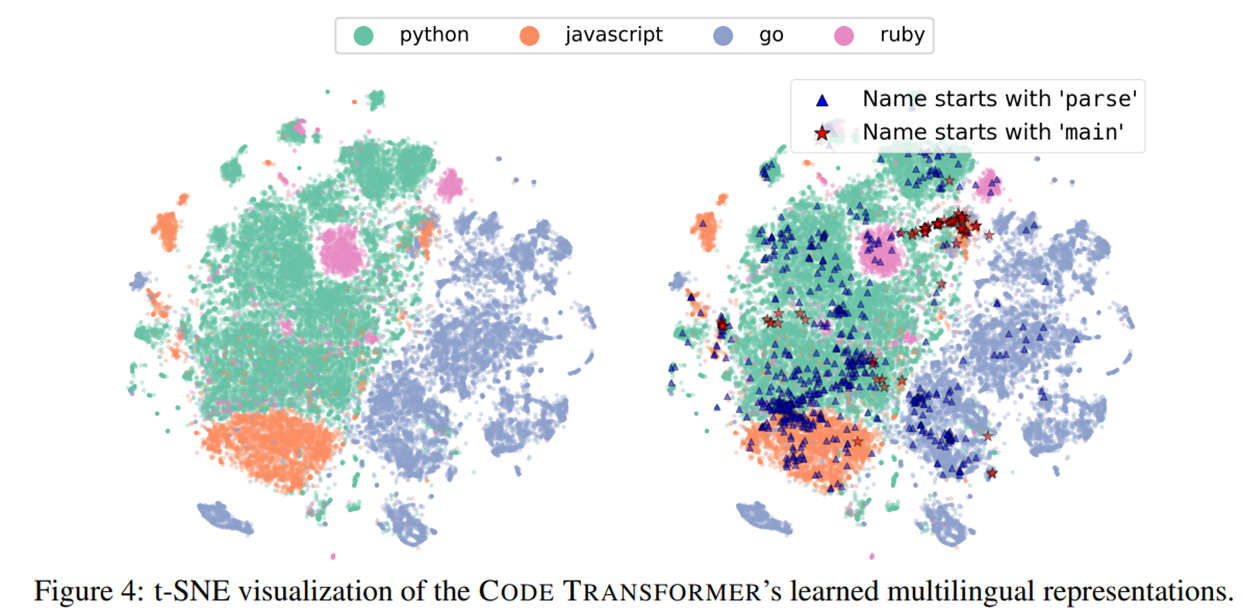
消融实验：分析指针网络、上下文和结构信息的重要性，实验结果表明在使用指针网络的情况下，上下文和结构信息相结合对模型的提升效果最为明显。





定性分析：

虽然来自同一语言的代码片段往往组合在一起，但不同编程语言之间存在有趣的交集。以parse开头的代码片段主要位于Python和Javascript的交集区域。



示例代码段以parse开始(左)及其与其他语言的最佳嵌入匹配(右)。这两种方法都会解析输入字符串，将其转换为布尔值。尽管它们在语义上非常相似，但它们的方法名称不同；它们在代码transformer编码器中的表示反映了它们的语义相似性。

