Deep Learning On Code with an Unbounded Vocabulary

A major challenge when using techniques from Natural Language Processing for supervised learning on computer program source code is that many words in code are neologisms. Reasoning over such an unbounded vocabulary is not something NLP methods are typically suited for. We introduce a deep model that contends with an unbounded vocabulary (at training or test time) by embedding new words as nodes in a graph as they are encountered and processing the graph with a Graph Neural Network.

当使用自然语言处理技术对计算机程序源代码进行监督学习时，一个主要挑战是代码中的许多单词都是新词。对这种无界词汇的推理不是 NLP 方法通常适合的。我们引入了一个深层模型，它与无界词汇相抗衡 (在训练或测试时间) 通过在遇到新词时将新词作为节点嵌入图形中，并使用图形神经网络处理图形。

1 Introduction

Computer program source code is an abundant, accessible, and important form of data. But despite the similarities between natural language and source code, deep learning methods for Natural Language Processing (NLP) have not been straightforward to apply to learning problems on source code like code completion and automated variable name generation.[1]

There are many reasons for this, but two central ones are:

1. Code is extremely sensitive to syntax. Natural language sentences can be messy and still get their point across. It is precisely this lack of rigid structure that makes learning necessary for understanding natural language in the first place. But this lack of structure makes source code a challenge for NLP methods: a tiny change in code syntax can result in a huge change in semantics. E.g. one tab too few in Python can completely change the contents of a for loop.

2. Code is written using an unbounded vocabulary. Natural language is mostly composed of words from a large but fixed vocabulary. Standard NLP methods can thus perform well by fixing a large vocabulary of words before training, labeling the few words they encounter outside this vocabulary as “Unknown”. But in code every new variable or method declared requires a new, often abstruse, name. A model must be able to reason about these neologisms to understand code.

计算机程序源代码是一种丰富的、可访问的、重要的数据形式。但是，尽管自然语言和源代码之间有相似之处，但自然语言处理(NLP)的深度学习方法并没有直接应用于像代码完成和自动变量名生成这样的源代码学习问题

造成这一现象的原因有很多，但其中两个主要原因是:

1. 代码对语法非常敏感。自然语言的句子可能很凌乱，但仍然能让人理解它们的意思。正是由于缺乏严格的结构，才使得学习成为理解自然语言的必要条件。但是这种结构的缺乏使得源代码对NLP方法来说是一个挑战:代码语法的微小变化可能导致语义的巨大变化。例如，Python中一个选项卡太少就可以完全改变for循环的内容。

1. 代码是使用无界词汇表编写的。自然语言主要由大量但固定的词汇组成。因此，标准的NLP方法可以在训练前固定大量词汇，并将这些词汇之外的少数词汇标记为“未知”，从而很好地执行任务。但是在代码中，每个新声明的变量或方法都需要一个新的，通常是抽象的名称。模型必须能够推理出这些新词来理解代码。