Evaluating Large Language Models Trained on Code -hoho

论文试图解决什么问题？

本文是CodeX的论文（即GitHub Copilot）, 解决代码生成问题。本文集中在用文档来生成Python的函数，并用新模型（CodeX）验证是否有更高的准确度。

这是否是一个新的问题？

不是新问题，本文论述了一些CodeX之前的代码生成模型

这篇文章要验证一个什么科学假设？

代码生成模型性能是否可以继续提升，本文用了GPT的模型

有哪些相关研究？如何归类？谁是这一课题在领域内值得关注的研究员？

相关研究有以下几类：

- 代码生成

当前有两种比较流行的智能代码学习方法：

1. 代码归纳，通过潜在的代码表示去生成代码
2. 代码合成，通常用自然语言的表达去生成代码。这一类很多跟生成AST有关。

一种比较经典的方法是使用PCFG(probabilistic context free grammar)生成AST（可参考作者Maddison & Tarlow）

也有直接通过n-gram，character-level语言模型合成代码。

然后到了使用大规模语言模型，如CodeBert，PyMT5

还有另外一些用不同寻常的方法：如非监督学习（TransCoder）,对比学习（ContraCode）

- 关于数据集的生成、生成质量验证方法

最近比较出名的benchmark有CodeXGLUE，APPS等

CodeSearchNet建立了多种流行的编程语言数据集

- 与生成代码相关的周边技术

如生成unit test（可参考作者Tufano，用transformer生成）

代码补全工具

代码缺陷定位与修复（可参考作者Drain ,使用神经翻译系统来修复代码）

论文中提到的解决方案之关键是什么？

使用GitHub上的代码fine-tune GPT。作者发现在pre-trained模型上fine-tune看不到模型有提升，但是在GPT(未经过训练的)上，模型收敛很快，于是决定用GPT。

论文中的实验是如何设计的？

定义度量：pass@k，k个样本里面有一个样本解决问题的比率。对于一个问题，采样k份代码，只要有1份代码解决这个问题（通过单元测试），即表明模型解决问题，然后就可以考察总的解决问题的数据量占比。

作者定义了一个无偏的版本：$pass@k = \mathbb{E}_{problems}[1 - \frac{C_{n-c}^k}{C_n^k}]$，对每个问题采样n个样本（$c \geq k$），计算正确的解题数量的样本c($c \leq n$)

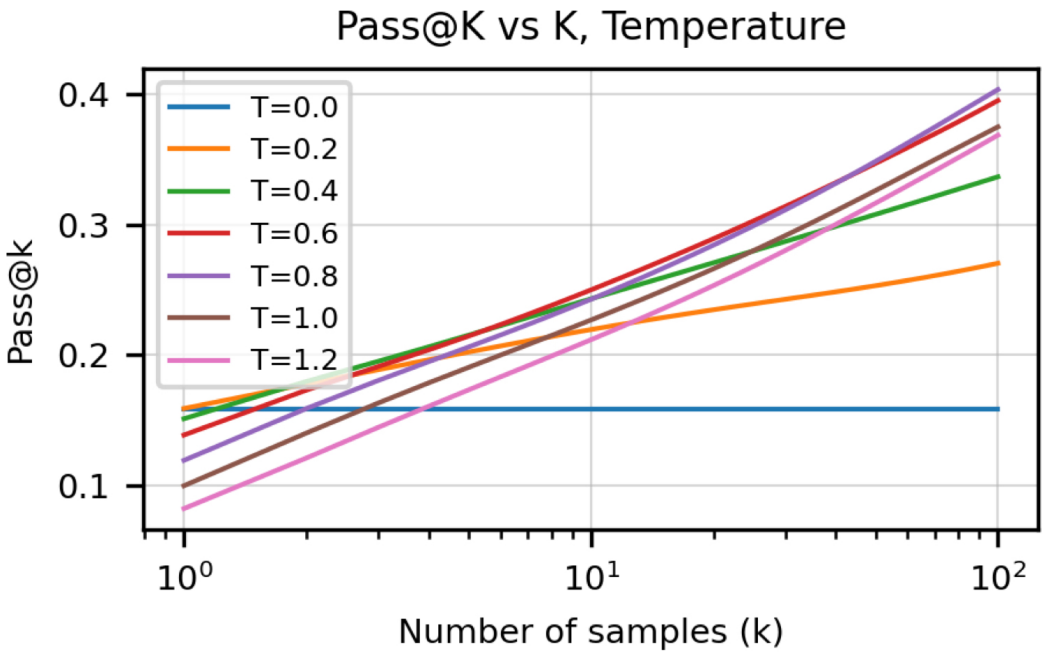
用于定量评估的数据集是什么？代码有没有开源？

用于评估的数据集https://github.com/openai/human-eval

代码没开源

论文中的实验及结果有没有很好地支持需要验证的科学假设？

由下图可见，随着采样数量与模型的pass@k几乎呈正相关关系。



本文还加了个temperature系数，发现大的temperature系数适用于大的k值（hoho_todo: temperature系数用在哪？）

本文还验证了该模型生成代码注释的能力，使用人工评估的方式。可能由于数据质量的问题（如程序员往往不乐意写注释，或写与代码内容不相关的东西），所以这一方面的评估表现平平。然后，本文又通过这种方法，用back-translation去采样生成代码的：只需最大化概率：

P(ground truth docstring | generated sample)，这里的P是生成代码注释的model。

这篇论文到底有什么贡献？

验证了使用大规模语言模型GPT，通过fine-tune，可以较好的生成代码。并且，该模型还可以简单通过训练即可完成反向任务：从代码到代码文档的生成

下一步呢？有什么工作可以继续深入？

本文列出了一些该模型的缺陷，可以往这些方面继续深入：

1. Misalignment

模型会生成错误的代码，因为错误的代码跟训练数据的分布是一样的。这就是一种alignment failure：系统有能力去正确完成，但它并没有这么做。

2. 数据偏见问题，模型会生成”有害“的代码