如何阅读别人的代码

挺多人问过我「如何阅读已有代码」这个问题，希望我能有一个好方法。有些人希望通过阅读「优质项目」（比如 Linux 内核）得到提高，改进自己的代码质量。对于这个问题，我一般都不好回答，因为我很少从阅读别人的代码得到提升。我对自己阅读的代码有很高的标准，因为世界上存在太多风格差劲的代码，阅读它们会损害自己的思维。同样的道理，我也不会阅读风格差劲的文章。

但这并不等于我无法跟其它程序员交流和共事，我有别的办法。比起阅读代码，我更喜欢别人给我讲解他们的代码，用简单的语言或者图形来解释他们的思想。有了思想，我自然知道如何把它变成代码，而且是优雅的代码。很多人的代码我不会去看，但如果他们给我讲，我是可以接受的。

如果有同事请我帮他改进代码，我不会拿起代码埋头就看，因为我知道看代码往往是事倍功半，甚至完全没用。我会让他们先在白板上给我解释那些代码是什么意思。我的同事们都发现，把我讲明白是需要费一番工夫的。因为我的要求非常高，只要有一点不明白，我就会让他们重新讲。还得画图，我会让他们反复改进画出来的图，直到我能一眼看明白为止。如果图形是 3D 的，我会让他们给我压缩成 2D 的，理解了之后再推广到 3D。我无法理解复杂的，高维度的概念，他们必须把它给我变得很简单。

所以跟我讲代码可能需要费很多时间，但这是值得的。我明白了之后，往往能挖出其他人都难以看清楚的要点。给我讲解事情，也能提升他们自己的思维和语言能力，帮助他们简化思想。很多时候我根本没看代码，通过给我讲解，后来他们自己就把代码给简化了。节省了我的脑力和视力，他们也得到了提高。

我最近一次看别人的代码是在 Intel，我们改了 PyTorch 的代码。那不是一次愉悦的经历，因为虽然很多人觉得 PyTorch 好用，它内部的代码却是晦涩而难以理解的。PyTorch 不是 Intel 自己的东西，所以没有人可以给我讲。修改 PyTorch 代码，增加新功能的时候，我发现很难从代码本身看明白应该改哪里。后来我发现，原因在于 PyTorch 的编译构架里自动生成了很多代码，导致你无法理解一些代码是怎么来的。

比如他们有好几个自己设计的文件格式，里面有一些特殊的文本，决定了如何在编译时生成代码。你得理解这些文件在说什么，而那不是任何已知的语言。这些文件被一些 Python 脚本读进去，吐出来一些奇怪的 C++，CUDA，或者 Python 代码。这其实是一种 DSL，我已经在之前的文章中解释过 DSL 带来的问题。要往 PyTorch 里面加功能，你就得理解这些脚本是如何处理这些 DSL，生成代码。而这些脚本写得也比较混乱和草率，所以就是头痛加头痛。

最后我发现，没有办法完全依靠这些代码本身来理解它。那么怎么解决这个问题呢？幸好，网络上有 PyTorch 的内部工程师写了篇 blog，解释 PyTorch 如何组织代码。Blog 的作者 E. Z. Yang 我见过一面，是在一次 PL 学术会议上。他当时在 MIT 读书，一个挺聪明的小伙子。不过看了这 blog 也只能初步知道它做了什么，应该碰大概哪些文件，而这些每天都可能变化。

这篇 blog 还提到，某几个目录里面是历史遗留代码，如果你不知道那是什么，那么请不要碰！看看那几个目录，里面都是一些利用 C 语言的宏处理生成代码的模板，而它使用 C 语言宏的方式还跟普通的用法不一样。在我看来，所谓「宏」（macro）和 「元编程」（metaprogramming） 本身就是巨大的误区，而 PyTorch 对宏的用法还如此奇怪，自作聪明。

你以为看了这篇 blog 就能理解 PyTorch 代码了吗？不，仍然是每天各种碰壁。大量的经验都来自折腾和碰壁。多个人同时在进行这些事情，然后分享自己的经验。讨论会内容经常是：「我发现要做这个，得在这个文件里加这个，然后在那个文件里加那个…… 然后好像就行了。」 下次开会又有人说：「我发现不是像你说的那样，还得改这里和这里，而那里不是关键……」 许多的知其然不知其所以然，盲人摸象，因为「所以然」已经被 PyTorch 的作者们掩盖在一堆堆混乱的 DSL 下面了。

所以我从 PyTorch 的代码里面学到了什么呢？什么都没有。我只看到各种软件开发的误区在反复上演。如果他们在早期得到我的建议，根本不可能把代码组织成这种样子，不可能有这么多的宏处理，代码生成，DSL。PyTorch 之类的深度学习框架，本质上是某种简单编程语言的解释器，只不过这些语言写出来的函数可以求导而已。

很多人都不知道，有一天我用不到一百行 Scheme 代码就写出了一个「深度学习框架」，它其实是一个小的编程语言。虽然没有性能可言，没有 GPU 加速，功能也不完善，但它抓住了 PyTorch 等大型框架的本质——用这个语言写出来的函数能自动求导。这种洞察力才是最关键的东西，只要抓住了关键，细节都可以在需要的时候琢磨出来。几十行代码反复琢磨，往往能帮助你看透上百万行的项目里隐藏的秘密。

很多人以为看大型项目可以提升自己，而没有看到大型项目不过是几十行核心代码的扩展，很多部分是低水平重复。几十行平庸甚至晦涩的代码，重复一万次，就成了几十万行。看那些低水平重复的部分，是得不到什么提升的。造就我今天的编程能力和洞察力的，不是几百万行的大型项目，而是小到几行，几十行之短的练习。不要小看了这些短小的代码，它们就是编程最精髓的东西。反反复复琢磨这些短小的代码，不断改进和提炼里面的结构，磨砺自己的思维。逐渐的，你的认识水平就超越了这些几百万行，让人头痛的项目。

所以我如何阅读别人的代码呢？Don’t。如果有条件，我就让代码的作者给我讲，而不是去阅读它。如果作者不合作，而我真的要使用那个项目的代码，我才会去折腾它。那么如何折腾别人的代码呢？我有另外一套办法。