

照着论文造一台

本课讲 2018 年的一篇教程。哈佛大学的 Alexander Rush 把第 13 课的 Transformer 论文一段一段拆开，每段后面配上能运行的代码。2022 年，Austin Huang 等人又更新了一版。

适合 10 岁左右

约 40 分钟

需要：铅笔、剪刀、3 张纸、两位同伴

1 热身：说明书漏了什么

下面 3 条说明，大人一看就懂。假如你从来没做过，照着做会卡在哪里？在横线上写下你想问的问题。

煮面：把面放进锅里，煮好就吃。 我会问：_____

画小猫：画一只坐着的小猫。 我会问：_____

洗手：把手洗干净。 我会问：_____

比如：要先放水吗？煮多久才算“好”？一句话谁都看得懂，**真照着做，问题就冒出来了**。论文也是这样。

2 看懂论文，和真做出来，是两回事

第 13 课讲了 2017 年提出 **Transformer** 的论文。它像一份“大厨的说明”：重要的事都写了，很多小步骤却省掉了。官方代码又一层套一层，初学者很难找到“注意力到底在哪一行”。

Rush 干脆自己从头写一遍，边写边讲。他发现，**论文里一句话，代码里往往要写很多行**：

论文里的一句话	写代码时要补上的事
写译文时，不能偷看后面的词	做一张阶梯形的“遮挡卡”，挡住后面的位置（见第 3 节的图）
把句子分成一批一批来练	句子有长有短，短的要补上空格；还要告诉电脑哪些是空格，别去学它们
别让电脑把话说太满	正确的词只给 9 成把握，剩下 1 成分给其他词

还有些细节，论文根本没写，比如遮挡卡做成什么形状。写代码的人只能自己想出来。

3 这篇教程的做法：带批注的课本

有的课本旁边写满了批注。这篇教程就是一本“带批注的论文”，按论文原来的顺序一段接一段排下去：

- ① 论文怎么说：写译文时，不能偷看后面的词。
- ② 代码怎么写：`mask = subsequent_mask(5)`
- ③ 说明：做一张 5 行 5 列的遮挡卡。第 3 行只露出前 3 格。

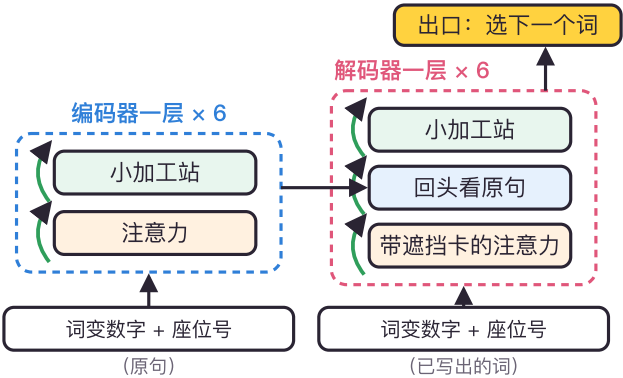
示意：一小段的样子。第 ② 条是一行真实的代码，不用看懂。右图就是这张遮挡卡：黄格子露出来，灰格子被挡住，红框是第 3 行。



读完最后一段，就有了一台能练习、能翻译的 Transformer。这种文字和代码交替着写的办法，叫**文学编程**，是计算机科学家 Knuth 提出的。

4 把 Transformer 拆成积木

每块积木干什么，第 13 课讲过。这里只看教程怎么**先做小积木，再拼成大机器**：



示意图：绿色弯线是近路。教程先做好一层，再**复制 6 份**叠起来，像乐高先拼一个房间，再复制成一栋楼。

积木	干什么（打个比方）	积木	干什么（打个比方）
词变数字	每个词换成一串数字，像一张“数字身份证”。	小加工站	每个词单独进去，加工一下再出来。
座位号	注意力本身不管先后。贴上座位号，才知道谁排第几。	近路和整理	近路是第 9 课讲的。整理是把一串数字调到差不多大小。

积木	干什么（打个比方）	积木	干什么（打个比方）
注意力	每个词看看其他词，找出最有关的。默认 8 个“头”，像 8 双眼睛。	编码器 / 解码器	一个读原句，一个写译文，各叠 6 层。解码器还多一块“回头看原句”。

一个有趣的发现：代码和论文不完全一样

论文是先走近路合流，再整理。代码换成先整理，再进积木。作者在说明里写了：这样代码更简单。这和第 10 课把整理站挪到小路上很像，后来的大模型大多也这样排。

5 造好了，先考一道“照抄题”

机器拼好了，教程没有马上拿去翻译。它先出了一道最简单的题：给它一串随机的数字，让它原样照抄出来。这个小测验叫复制任务。

题目	1 5 3 8 2 7
刚开始	4 4 4 1 9 4
练了一会儿	1 5 3 3 2 2
练好了	1 5 3 8 2 7 ✓

示意：数字是我们编的，不是教程里的真实输出。

照抄题不用下载任何资料，几分钟就能练好。座位号没接好，顺序就会乱。遮挡卡放错了，机器就会偷看答案。所以能抄对，说明零件都接对了。就像新自行车先在院子里骑一圈，再上马路。通过以后，教程才做真任务：把德语翻译成英语。

6 “看懂”的最好办法：自己做一遍

读论文时，人常常觉得“差不多懂了”。一动手，坑就冒出来：数字排列的形状对不上，遮挡卡放错地方。这篇教程把这些坑提前标了出来，成了学 Transformer 的标准材料。

懂有四级：读过一遍 → 能讲给别人听 → 能照着做出来 → 做坏了能自己修好。越往后，懂得越深。

7 这和今天有什么关系

今天的聊天 AI，大脑结构大多是 Transformer 的后代。它们一般只用解码器那一叠，一个词一个词往后写。很多人学 Transformer 时还会读这份代码。不过，真正的聊天 AI 大得多，零件也换成了更快的版本。这份代码是“教学模型”。它留下的最大财富是一种学习方法：边读边做，再用最简单的题检查每个零件。

8 动手：把菜谱翻译成步骤

玩法（2 个人，约 15 分钟）

1. 菜谱卡：“果酱面包：拿两片面包，抹上果酱，合起来就好了。”A 把它拆成步骤，写进下表，**每步只做一个动作**。
2. B 当“机器人”，只照字面做，没写的不做。两本书当面包，尺子当小刀，空盒子当果酱瓶。
3. 机器人卡住或做错，就记下来。A 补上一步，从头再做。换人再拆“纸飞机：对折，折出尖头，就能飞了”。

第几步	我写的步骤	机器人出了什么问题
1		
2		
3		

1 句话拆成了 _____ 步。**你会发现：**“打开瓶盖”这种事菜谱没写，机器人却不知道。论文变代码，也要补上没写的步骤。

9 动手：纸上 Transformer 和照抄测试

第一步：拼积木（1 个人，约 5 分钟）

剪 10 张卡片，照第 4 节积木图里的 10 个名字写好（小加工站写 2 张）。打乱后**合上讲义**，自己从下往上拼，再翻开检查。

第二步：照抄测试（3 个人，约 10 分钟）

1. **出题员**想 5 个数字，每个写在一张小纸片上，再抄进下表。
2. **座位员**在纸片背面写座位号 1—5。第 1 轮他“坏了”，什么都不写。
3. **注意力员**把纸片打乱再传下去。注意力本身不管顺序，“打乱”就是模拟这一点。
4. **解码员**只看纸片，按第 1 位到第 5 位写出来。

轮次	座位员	题目	解码员写出的	抄对了吗
1	坏了			
2	正常			

你会发现：少了座位号，数字一个不少，顺序却乱了。照抄题能查出哪块积木没接好。

10 词语小卡片

复现

照着论文，自己把东西从头做出来，看看结果对不对。

文学编程

文字和代码交替着写。程序按人读的顺序排，一边讲，一边写。

遮挡卡（掩码）

写第几个词，就只露出前几个位置。不让机器偷看后面的答案。

照抄测试（复制任务）

让新造的机器原样抄一串数字。能抄对，说明零件大体接对了。

11 想一想

1. 论文为什么不把每个小步骤都写出来？如果全写出来，会有什么麻烦？

2. 为什么先考“照抄”再做翻译？直接翻译翻错了，能找到哪里坏了吗？

3. 你学过的一样本领（比如骑车、做手工、玩魔方），是“看懂”的，还是“做会”的？两者有什么不同？

一句话记住：论文一句话，代码可能是一大段；自己动手造一遍，才算真看懂。

给家长和老师：本课讲的是 Harvard NLP 的 [The Annotated Transformer](#)。原版由 Alexander M. Rush 于 2018 年 4 月发布 (NLP-OSS 2018)，2022 年版由 Austin Huang、Suraj Subramanian、Jonathan Sum 等人更新。它按 Attention Is All You Need 原文顺序，每段后紧跟 PyTorch 实现，补上掩码、批处理、学习率调度、标签平滑等细节。代码的 SublayerConnection 是归一化前置，论文是后置 $\text{norm}(x + \text{sublayer}(x))$ 。例子先做随机符号的复制任务，再做 Multi30k 德译英。

讲义中“论文里的一句话”是通俗改写；比喻、积木图、照抄数字和两个活动都是示意，“打乱纸片”只模拟注意力不区分顺序。详见本站 [第 14 天中文导读](#)和 [30u30 第 14 天目录](#)（英文）。