

照著論文造一臺

本課講 2018 年的一篇教程。哈佛大學的 Alexander Rush 把第 13 課的 Transformer 論文一段一段拆開，每段後面配上能運行的代碼。2022 年，Austin Huang 等人又更新了一版。

適合 10 歲左右

約 40 分鐘

需要：鉛筆、剪刀、3 張紙、兩位同伴

1 熱身：說明書漏了什麼

下面 3 條說明，大人一看就懂。假如你從來沒做過，照著做會卡在哪裡？在橫線上寫下你想問的問題。

煮麵：把面放進鍋裡，煮好就吃。 我會問：_____

畫小貓：畫一隻坐著的小貓。 我會問：_____

洗手：把手洗乾淨。 我會問：_____

比如：要先放水嗎？煮多久才算「好」？一句話誰都看得懂，**真照著做，問題就冒出來了**。論文也是這樣。

2 看懂論文，和真做出來，是兩回事

第 13 課講了 2017 年提出 **Transformer** 的論文。它像一份「大廚的說明」：重要的事都寫了，很多小步驟卻省掉了。官方代碼又一層套一層，初學者很難找到「注意力到底在哪一行」。

Rush 乾脆自己從頭寫一遍，邊寫邊講。他發現，**論文裡一句話，代碼裡往往要寫很多行**：

論文裡的一句話	寫代碼時要補上的事
寫譯文時，不能偷看後面的詞	做一張階梯形的「遮擋卡」，擋住後面的位置（見第 3 節的圖）
把句子分成一批一批來練	句子有長有短，短的要補上空格；還要告訴電腦哪些是空格，別去學它們
別讓電腦把話說太滿	正確的詞只給 9 成把握，剩下 1 成分給其他詞

還有些細節，論文根本沒寫，比如遮擋卡做成什麼形狀。寫代碼的人只能自己想出來。

3 這篇教程的做法：帶批註的課本

有的課本旁邊寫滿了批註。這篇教程就是一本「帶批註的論文」，按論文原來的順序一段接一段排下去：

- ① 論文怎麼說：寫譯文時，不能偷看後面的詞。
- ② 代碼怎麼寫：`mask = subsequent_mask(5)`
- ③ 說明：做一張 5 行 5 列的遮擋卡。第 3 行只露出前 3 格。

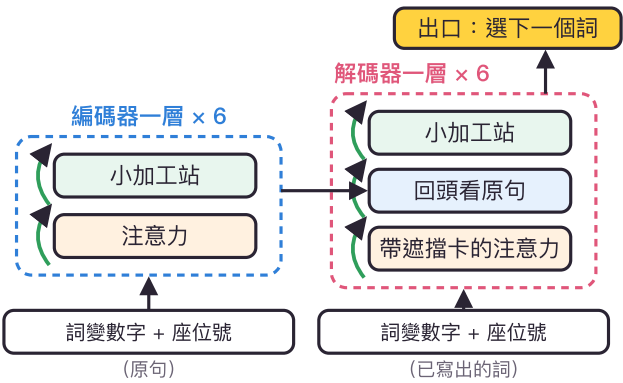
示意：一小段的樣子。第 ② 條是一行真實的代碼，不用看懂。右圖就是這張遮擋卡：黃格子露出來，灰格子被擋住，紅框是第 3 行。



讀完最後一段，就有了一臺能練習、能翻譯的 Transformer。這種文字和代碼交替著寫的辦法，叫**文學編程**，是計算機科學家 Knuth 提出的。

4 把 Transformer 拆成積木

每塊積木幹什麼，第 13 課講過。這裡只看教程怎麼**先做小積木，再拼成大機器**：



示意圖：綠色彎線是近路。教程先做好一層，再**複製 6 份**疊起來，像樂高先拼一個房間，再複製成一棟樓。

積木	幹什麼（打個比方）	積木	幹什麼（打個比方）
詞變數字	每個詞換成一串數字，像一張「數字身份證」。	小加工站	每個詞單獨進去，加工一下再出來。
座位號	注意力本身不管先後。貼上座位號，才知道誰排第幾。	近路和整理	近路是第 9 課講的。整理是把一串數字調到差不多大小。

積木	幹什麼（打個比方）	積木	幹什麼（打個比方）
注意力	每個詞看看其他詞，找出最有關的。默認 8 個「頭」，像 8 雙眼睛。	編碼器 / 解碼器	一個讀原句，一個寫譯文，各疊 6 層。解碼器還多一塊「回頭看原句」。

一個有趣的發現：代碼和論文不完全一樣

論文是先走近路合流，**再整理**。代碼換成**先整理**，再進積木。作者在說明裡寫了：這樣代碼更簡單。這和第 10 課把整理站挪到小路上很像，後來的大模型大多也這樣排。

5 造好了，先考一道「照抄題」

機器拼好了，教程沒有馬上拿去翻譯。它先出了一道最簡單的題：**給它一串隨機的數字，讓它原樣照抄出來**。這個小測驗叫**複製任務**。

題目	1 5 3 8 2 7
剛開始	4 4 4 1 9 4
練了一會兒	1 5 3 3 2 2
練好了	1 5 3 8 2 7 ✓

示意：數字是我們編的，不是教程裡的真實輸出。

照抄題不用下載任何資料，幾分鐘就能練好。座位號沒接好，順序就會亂。遮擋卡放錯了，機器就會偷看答案。所以**能抄對，說明零件都接對了**。就像新自行車先在院子裡騎一圈，再上馬路。通過以後，教程才做真任務：**把德語翻譯成英語**。

6 「看懂」的最好辦法：自己做一遍

讀論文時，人常常覺得「差不多懂了」。一動手，坑就冒出來：數字排列的形狀對不上，遮擋卡放錯地方。這篇教程**把這些坑提前標了出來**，成了學 Transformer 的標準材料。

懂有四級：**讀過一遍 → 能講給別人聽 → 能照著做出來 → 做壞了能自己修好**。越往後，懂得越深。

7 這和今天有什麼關係

今天的聊天 AI，大腦結構大多是 Transformer 的後代。它們一般只用**解碼器那一疊**，一個詞一個詞往後寫。很多人學 Transformer 時還會讀這份代碼。不過，真正的聊天 AI 大得多，零件也換成了更快的版本。這份代碼是「教學模型」。它留下的最大財富是一種**學習方法**：邊讀邊做，再用最簡單的題檢查每個零件。

8 動手：把菜譜翻譯成步驟

玩法（2 個人，約 15 分鐘）

1. 菜譜卡：「果醬麵包：拿兩片面包，抹上果醬，合起來就好了。」A 把它拆成步驟，寫進下表，每步只做一個動作。
2. B 當「機器人」，只照字面做，沒寫的不做。兩本書當面包，尺子當小刀，空盒子當果醬瓶。
3. 機器人卡住或做錯，就記下來。A 補上一步，從頭再做。換人再拆「紙飛機：對摺，折出尖頭，就能飛了」。

第幾步	我寫的步驟	機器人出了什麼問題
1		
2		
3		

1 句話拆成了 _____ 步。你會發現：「打開瓶蓋」這種事菜譜沒寫，機器人卻不知道。論文變代碼，也要補上沒寫的步驟。

9 動手：紙上 Transformer 和照抄測試

第一步：拼積木（1 個人，約 5 分鐘）

剪 10 張卡片，照第 4 節積木圖裡的 10 個名字寫好（小加工站寫 2 張）。打亂後合上講義，自己從下往上拼，再翻開檢查。

第二步：照抄測試（3 個人，約 10 分鐘）

1. 出題員想 5 個數字，每個寫在一張小紙片上，再抄進下表。
2. 座位員在紙片背面寫座位號 1—5。第 1 輪他「壞了」，什麼都不寫。
3. 注意力員把紙片打亂再傳下去。注意力本身不管順序，「打亂」就是模擬這一點。
4. 解碼員只看紙片，按第 1 位到第 5 位寫出來。

輪次	座位員	題目	解碼員寫出的	抄對了嗎
1	壞了			
2	正常			

你會發現：少了座位號，數字一個不少，順序卻亂了。照抄題能查出哪塊積木沒接好。

10 詞語小卡片

復現

照著論文，自己把東西從頭做出來，看看結果對不對。

文學編程

文字和代碼交替著寫。程序按人讀的順序排，一邊講，一邊寫。

遮擋卡（掩碼）

寫第幾個詞，就只露出前幾個位置。不讓機器偷看後面的答案。

照抄測試（複製任務）

讓新造的機器原樣抄一串數字。能抄對，說明零件大體接對了。

11 想一想

1. 論文為什麼不把每個小步驟都寫出來？如果全寫出來，會有什麼麻煩？

2. 為什麼先考「照抄」再做翻譯？直接翻譯翻錯了，能找到哪裡壞了嗎？

3. 你學過的一樣本領（比如騎車、做手工、玩魔方），是「看懂」的，還是「做會」的？兩者有什麼不同？

一句話記住：論文一句話，代碼可能是一大段；自己動手造一遍，才算真看懂。

給家長和老師：本課講的是 Harvard NLP 的 [The Annotated Transformer](#)。原版由 Alexander M. Rush 於 2018 年 4 月發佈（NLP-OSS 2018），2022 年版由 Austin Huang、Suraj Subramanian、Jonathan Sum 等人更新。它按 Attention Is All You Need 原文順序，每段後緊跟 PyTorch 實現，補上掩碼、批處理、學習率調度、標籤平滑等細節。代碼的 SublayerConnection 是歸一化前置，論文是後置 $\text{norm}(x + \text{sublayer}(x))$ 。例子先做隨機符號的複製任務，再做 Multi30k 德譯英。

講義中「論文裡的一句話」是通俗改寫；比喻、積木圖、照抄數字和兩個活動都是示意，「打亂紙片」只模擬注意力不區分順序。詳見本站 [第 14 天中文導讀](#)和 [30u30 第 14 天目錄](#)（英文）。