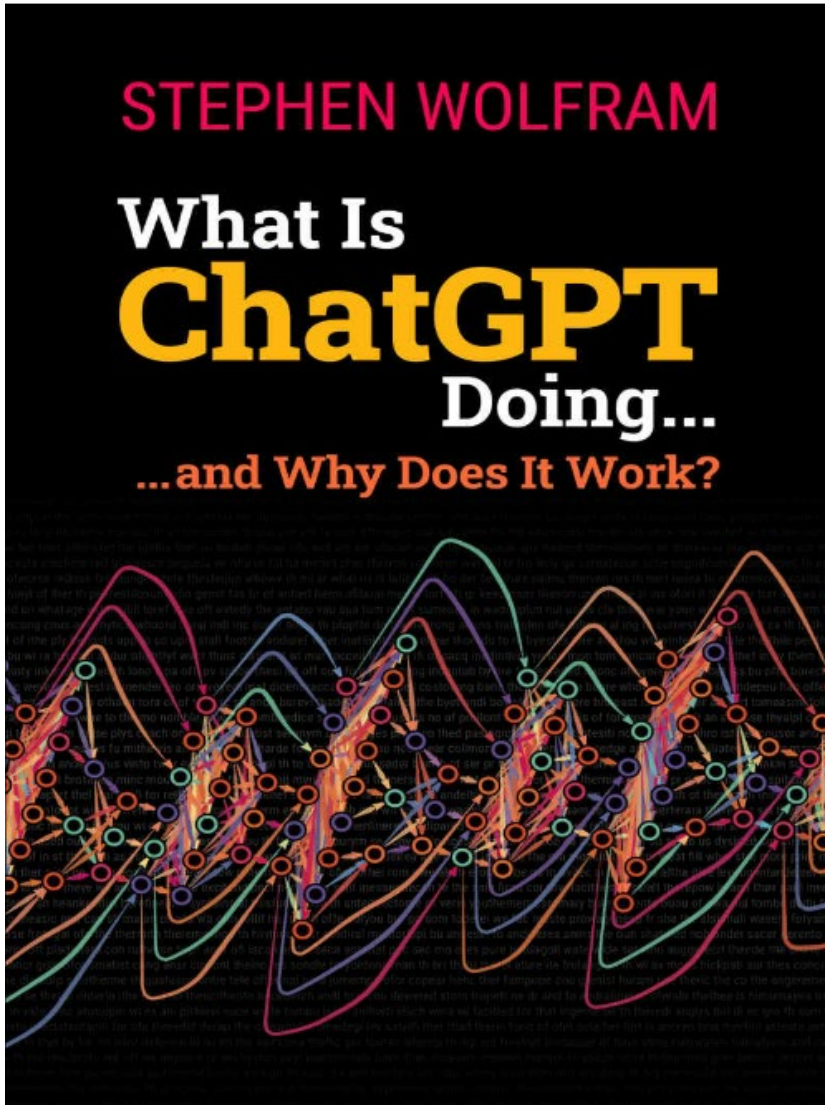


ChatGPT 在做什么...为什么它能做到?



作者：斯蒂芬·沃尔夫勒姆，翻译：ChatGPT，编辑：老皮

Table of Contents

| | |
|---|----|
| 前言 | 3 |
| ChatGPT 在做什么...为什么它能做到? | 4 |
| 它只是一次添加一个单词..... | 4 |
| 概率从何而来? | 8 |
| 什么是模型? | 11 |
| 人类式任务的模型 | 13 |
| 神经网络 | 14 |
| 机器学习和神经网络的训练..... | 22 |
| 神经网络训练的实践和经验..... | 26 |
| “当然，只要神经网络足够大，就能做任何事情！” | 30 |
| 嵌入（Embeddings）的概念..... | 32 |
| ChatGPT 内部 | 36 |
| ChatGPT 的训练 | 42 |
| 除了基本的训练之外 | 43 |
| 什么让 ChatGPT 能够工作? | 44 |
| 意义空间与语义运动规律..... | 48 |
| 语义语法和计算语言的力量..... | 52 |
| 那么.....ChatGPT 在做什么，它为什么有效呢? | 54 |
| 感谢 | 55 |
| Wolfram Alpha 是将计算知识超能力带给 ChatGPT 的途径 | 56 |
| ChatGPT 和 Wolfram Alpha | 56 |
| 一个基本的例子 | 57 |
| 更多例子 | 59 |
| 未来之路 | 69 |
| Additional Resources | 72 |
| Online link to all resources:..... | 72 |

前言

这本小书试图从基本原理解释 ChatGPT 的工作原理和原因。在某些方面，这是关于技术的故事。但它也是一部关于科学和哲学的故事。为了讲述这个故事，我们必须汇集数个世纪以来许多惊人的思想和发现。

对我来说，看到我长期以来感兴趣的许多事情在突然的进步中汇聚在一起是令人兴奋的。从简单程序的复杂行为到语言和意义的核心特征，以及大型计算机系统的实际应用，所有这些都是 ChatGPT 故事的一部分。

ChatGPT 基于神经网络的概念，神经网络最初是在 1940 年代作为大脑运作的理想化而发明的。我自己在 1983 年首次编写了一个神经网络，但它并没有做任何有趣的事情。但是，40 年后，随着计算机的速度快了百万倍，互联网上有数十亿页的文本，并且进行了一系列工程创新，情况已经完全不同了。出人意料的是，一个比我 1983 年使用的神经网络大十亿倍的神经网络能够生成有意义的人类语言，这曾被认为是独特的人类特征。

这本书包括我在 ChatGPT 发布后不久写的两篇文章。第一篇是关于 ChatGPT 及其生成语言能力的解释。第二篇展望了 ChatGPT 能够使用计算工具超越人类所能做的事情，特别是能够利用我们的 Wolfram|Alpha 系统的计算知识“超能力”。

ChatGPT 发布才三个月，我们刚刚开始了解它的实际和知识上的影响。但现在它的到来提醒我们，即使在已经发明和发现了所有东西之后，仍然有可能出现意外。



斯蒂芬·沃尔夫勒姆

2023 年 2 月 28 日

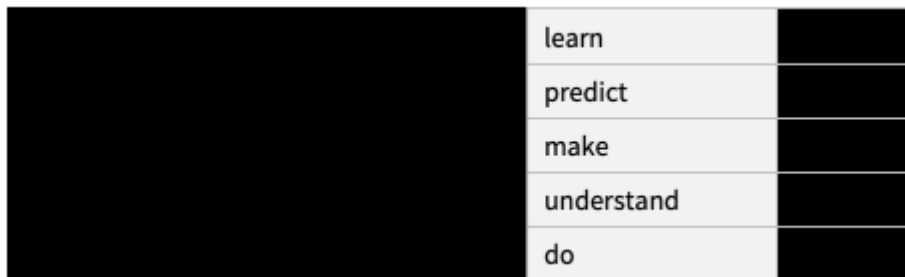
ChatGPT 在做什么...为什么它能做到?

它只是一次添加一个单词

ChatGPT 能够自动生成类似于人类书写的文本，这是非常惊人和意想不到的。但是它是如何做到的？它为什么能够如此出色地产生我们可能认为是有意义的文本呢？我的目的在于概述 ChatGPT 内部正在发生的事情，然后探索为什么它能够很好地产生文本。我首先要说明的是，我将重点关注正在发生的大局面，并且虽然我会提及一些工程细节，但我不会深入探讨它们。（而我将要说的实质内容同样适用于其他当前的“大型语言模型”[LLM]，而不仅仅是 ChatGPT。）

首先需要解释的是，ChatGPT 始终基本上是试图产生当前文本的“合理续写”，其中“合理”意味着“在看到人们在数十亿个网页上写了什么后，人们可能期望某人写下什么”。

假设我们有文本“AI 最好的事情是它的能力...”。想象一下扫描数十亿页人类编写的文本（例如网络上和数字化的书籍），找到所有这种文本的实例，然后看看接下来的单词在多大程度上会出现。ChatGPT 实际上会做类似的事情，除了（如我将解释的那样）它不会看字面文本；它寻找在某种意义上“匹配”的东西。但最终结果是，它会生成一个可能的单词排名列表，以及“概率”：



| | |
|------------|--|
| learn | |
| predict | |
| make | |
| understand | |
| do | |

值得注意的是，当 ChatGPT 写论文时，它实际上在一遍又一遍地询问“在给定的文本中，下一个单词应该是什么？”--每次添加一个单词。（更准确地说，如我将要解释的那样，它正在添加一个“令牌”，这可能只是单词的一部分，这就是为什么它有时可以“创造新词”的原因。）


但是，在每一步中，它都会得到一个具有概率的单词列表。但它应该选择哪个单词添加到它正在编写的文章（或其他内容）中呢？有人可能认为应该选择“排名最高”的单词（即被赋予最高“概率”的单词）。但是这里开始涉及到一些迷信。因为出于某种原因（也许有一天我们会以科学的方式理解），如果我们总是选择排名最高的单词，我们通常会得到一个非常“平淡”的文章，看起来从未“展示任何创造力”（甚至有时会逐字重复）。但是，如果有时（随机地）选择排名较低的单词，我们会得到一个“更有趣”的文章。

这里有随机性意味着，如果我们多次使用相同的提示，我们很可能每次都会得到不同的文章。与迷信的想法保持一致，这里有一个特殊的所谓“温度”参数，它确定了使用排名

较低的单词的频率。对于文章生成，事实证明，“温度”为 0.8 是最好的选择。（值得强调的是，这里没有使用任何“理论”；这只是实践中发现的有效方法。例如，“温度”概念的存在是因为在使用指数分布时，统计物理学中熟悉的概念恰好被使用了，但我们至少目前还不知道有什么“物理学”联系。）

在我们继续之前，我应该解释一下，为了说明方便，我大多数情况下不会使用 ChatGPT 中的完整系统；相反，我通常会使用一个更简单的 GPT-2 系统，其良好特性是足够小，可以在标准桌面计算机上运行。因此，对于我展示的所有内容，我都能够包含明确的 Wolfram 语言代码，您可以立即在计算机上运行。（单击此处的任何图片即可复制其后面的代码。）

例如，这是如何获取上面的概率表格。首先，我们必须检索底层的“语言模型”神经网络：

```
In[ ]:= model =  
      NetModel [ {"GPT2 Transformer Trained on WebText Data",  
                "Task" -> "LanguageModeling" } ]  
  
Out[ ]:= NetChain [  Input port: string  
                  Output port: class ]
```

稍后，我们将查看这个神经网络的内部，并讨论它是如何工作的。但是现在我们只需将这个“网络模型”应用为黑盒子来处理我们迄今为止的文本，并询问模型认为应该跟随的概率最高的前 5 个单词：

```
In[ ]:= [ " ", " ", " ", " ", " " ]  
Out[ ]:=
```

这将结果转换为明确格式化的“数据集”：

```
In[ ]:= [ [ [ ] ],  
          ( [ , ] ) ]  
  
Out[ ]:=  


|            |  |
|------------|--|
| learn      |  |
| predict    |  |
| make       |  |
| understand |  |
| do         |  |


```

这里展示了如果重复“应用模型”的结果-在每个步骤中添加具有最高概率的单词。

(在这个代码中指定为模型的“决策”)：

```
In[ ]:= [ , [ , " " ] ] ,  
"  
", ]  
  
Out[ ]:=
```

如果继续进行下去会发生什么？在这种“零温度”的情况下，很快就会变得混乱和重复：

The best thing about AI is its ability to learn from experience. It's not just a matter of learning from experience, it's learning from the world around you. The AI is a very good example of this. It's a very good example of how to use AI to improve your life. It's a very good example of how to use AI to improve your life. The AI is a very good example of how to use AI to improve your life. It's a very good example of how to use AI to

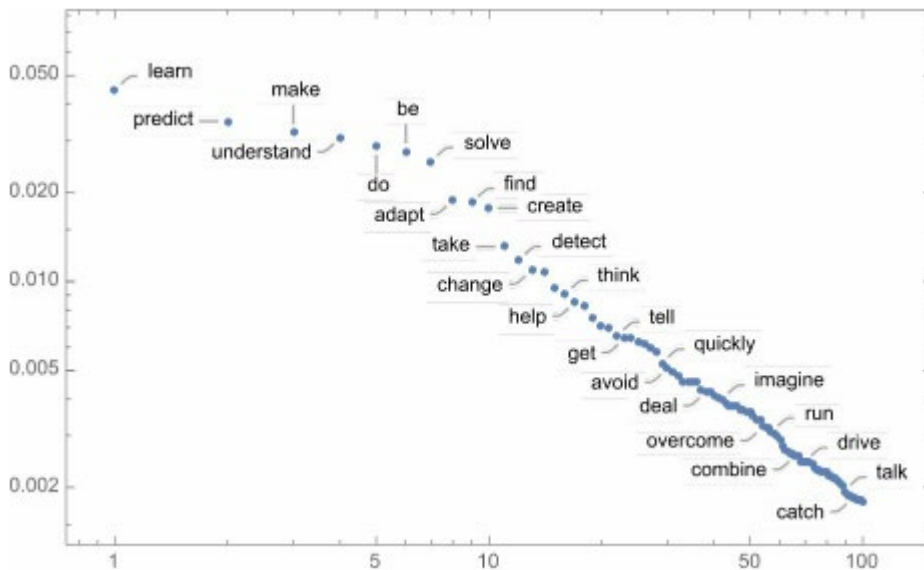
但是，如果不总是选择“顶部”单词，而是有时随机选择“非顶部”单词（与“随机性”相对应的“温度”为 0.8），又会发生什么呢？同样，我们可以逐步构建文本：

```
{ The best thing about AI is its ability to,  
The best thing about AI is its ability to create,  
The best thing about AI is its ability to create worlds,  
The best thing about AI is its ability to create worlds that,  
The best thing about AI is its ability to create worlds that are,  
The best thing about AI is its ability to create worlds that are both,  
The best thing about AI is its ability to create worlds that are both exciting,  
The best thing about AI is its ability to create worlds that are both exciting, }
```

每次这样做时，都会进行不同的随机选择，文本也会不同，就像这 5 个例子：



值得指出的是，即使在第一步，也有很多可能的“下一个单词”可供选择（在温度为 0.8 的情况下），尽管它们的概率很快就会下降（是的，在这个对数-对数图上的直线对应于一个 $n-1$ 的“幂律”衰减，这是语言的一般统计特征）：



那么如果继续进行下去会发生什么？这是一个随机的例子。它比顶部单词（零温度）的情况要好一些，但仍然最多有点奇怪：

The best thing about AI is its ability to see through, and make sense of, the world around us, rather than panicking and ignoring. This is known as AI "doing its job" or AI "run-of-the-mill." Indeed, taking an infinite number of steps, developing a machine that can be integrated with other systems, or controlling one system that's truly a machine, is one of the most fundamental processes of AI. Aside from the human-machine interaction, AI was also a big part of creativity

这是用最简单的 GPT-2 模型（来自 2019 年）完成的。使用更新的和更大的 GPT-3 模型会产生更好的结果。这是使用相同“提示”生成的顶部单词（零温度）文本，但使用最大的 GPT-3 模型：

The best thing about AI is its ability to automate processes and make decisions quickly and accurately. AI can be used to automate mundane tasks, such as data entry, and can also be used to make complex decisions, such as predicting customer behavior or analyzing large datasets. AI can also be used to improve customer service, as it can quickly and accurately respond to customer inquiries. AI can also be used to improve the accuracy of medical diagnoses and to automate the process of drug discovery.

这是一个“温度为 0.8”的随机例子：

The best thing about AI is its ability to learn and develop over time, allowing it to continually improve its performance and be more efficient at tasks. AI can also be used to automate mundane tasks, allowing humans to focus on more important tasks. AI can also be used to make decisions and provide insights that would otherwise be impossible for humans to figure out.

概率从何而来？

好的，ChatGPT 总是基于概率选择下一个单词。但是这些概率从哪里来？让我们从一个更简单的问题开始。让我们考虑每次生成一个英语字母（而不是单词）。我们如何确定每个字母的概率？

我们可以做的最简单的事情是只需获取英语文本的样本，并计算其中不同字母出现的频率。因此，例如，这计算了“cats”维基百科文章中的字母：

```
In[ ]:= LetterCounts [WikipediaData ["cats" ] ]  
  
<| e → 4279, a → 3442, t → 3397, i → 2739, s → 2615, n → 2464, o → 2426,  
Out[ ]:= r → 2147, h → 1613, l → 1552, c → 1405, d → 1331, m → 989, u → 916,  
f → 760, g → 745, p → 651, y → 591, b → 511, w → 509, v → 395, k → 212,  
T → 114, x → 85, A → 81, C → 81, l → 68, S → 55, F → 42, z → 38, E → 36
```

这样做同样适用于“dogs”：

```
In[ ]:= LetterCounts [WikipediaData ["dogs" ] ]  
  
<| e → 3911, a → 2741, o → 2608, i → 2562, t → 2528, s → 2406,  
Out[ ]:= n → 2340, r → 1866, d → 1584, h → 1463, l → 1355, c → 1083, g → 929,  
m → 859, u → 782, f → 662, p → 636, y → 500, b → 462, w → 409,  
v → 406, k → 151, T → 90, C → 85, l → 80, A → 74, x → 71, S → 65,
```

结果是相似的，但不完全相同（毫无疑问，“o”在“dogs”文章中更常见，因为它毕竟出现在“dog”这个词中）。不过，如果我们获取足够大的英语文本样本，我们可以期望最终获得至少相当一致的结果：

```
In[ ]:= English LANGUAGE [ character frequencies ]  
  
Out[ ]:= { e → 12.7%, t → 9.06%, a → 8.17%, o → 7.51%, i → 6.97%, n → 6.75%,  
s → 6.33%, h → 6.09%, r → 5.99%, d → 4.25%, l → 4.03%, c → 2.78%, u → 2.76%,  
m → 2.41%, w → 2.36%, f → 2.23%, g → 2.02%, y → 1.97%, p → 1.93%, b → 1.49%,  
v → 0.978%, k → 0.772%, j → 0.153%, x → 0.150%, q → 0.0950%, z → 0.0740% }
```

以下是我们仅仅根据这些概率生成字母序列的样本：

```
rronoitadatcaeaesaotdoysaroyiinnbantoioestlhddeocneoowceseciselnodrtrdgriscsatsepescnio:  
uhoetsedeyhedslernevstothindtbnnaohngotannbthrdthtonsiieldn
```

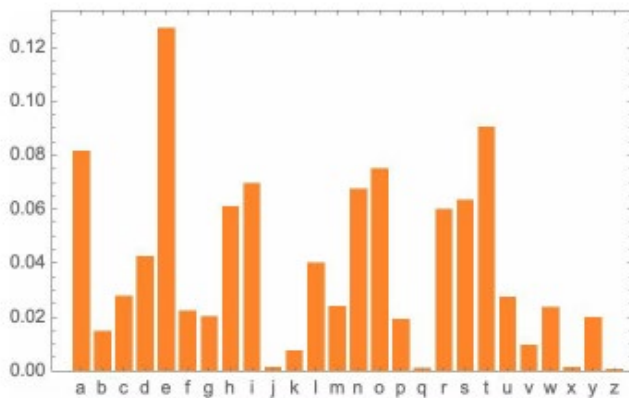

我们可以通过添加空格，将其视为具有一定概率的字母，将其分成“单词”：

我们可以通过强制“单词长度”的分布与英语中的分布相符来更好地生成“单词”：

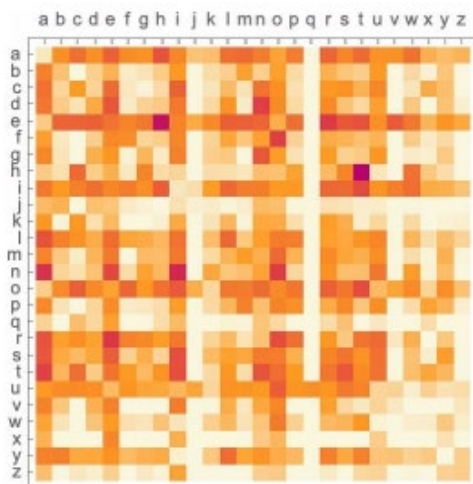
**sd n oeiaim satnwhoo eer rtr ofiiandrenapwokom del oaas ill e h f
rellptohltoettseodtrncilntehtotrkrhslo hdaol n sriaefr htthehtn ld gpod a h y oi**

我们这里没有得到任何“真实的单词”，但结果看起来稍微好一些了。但是，要进一步改进，我们需要做的不仅仅是随机选择每个字母。例如，我们知道，如果我们有一个“q”，下一个字母基本上必须是“u”。

这是字母概率的图：



这是显示典型英语文本中字母对 (“2-grams”) 的概率的图。第一个可能的字母显示在页面上，第二个字母向下显示：



在这里，我们看到“q”列除了在“u”行上以外，其他地方都是空白（概率为零）。好的，现在，我们不是选择每个字母。取而代之的是，让我们一次查看两个字母，并使用这些“2-gram”概率生成我们的“单词”。这是一个结果示例，其中包括一些“真实的单词”：

on inguman men ise forerenoft weat iofobato buc ous corew ousesetiv
falle tinouco ryefo ra the ecederi pasuthrgr cuconom tra tesla wil tat pere thi

通过足够多的英语文本，我们不仅可以获得单个字母或字母对（2-gram）的概率估计，还可以获得更长的字母序列的概率估计。如果我们使用逐步更长的 n-gram 概率生成“随机单词”，我们会发现它们逐渐变得“更真实”：

| | |
|---|--|
| 0 | |
| 1 | |
| 2 | |
| 3 | |
| 4 | |
| 5 | |

但是，让我们现在假设——大致如 ChatGPT 所做的那样——我们正在处理整个单词，而不是字母。在英语中，大约有 40,000 个常用词。通过查看大量的英语文本（例如几百万本书，总共有几千亿个单词），我们可以估计出每个单词的常见程度。使用这个，我们可以开始生成“句子”，其中每个单词都是独立随机选择的，其概率与其在语料库中出现的概率相同。这是一个结果示例：

of program excessive been by was research rate not here of of other is men
were against are show they the different the half the the in any were leaved

毫不奇怪，这是无意义的。那么，我们该如何做得更好呢？就像对于字母一样，我们可以开始考虑不仅仅是单词的概率，而是对单词对或更长的 n-gram 单词序列的概率进行考虑。对于单词对，这里是 5 个例子，它们都以“cat”单词作为起点：

| |
|--|
| |
| |
| |
| |
| |

看起来稍微“更有意义”。我们可以想象，如果我们能够使用足够长的 n-gram，我们基本上可以“获得一个 ChatGPT”——这意味着我们会获得一些会生成“具有正确整体文本概率”的文章长度的单词序列。但问题在于：没有足够的英语文本可用于推断这些概率。

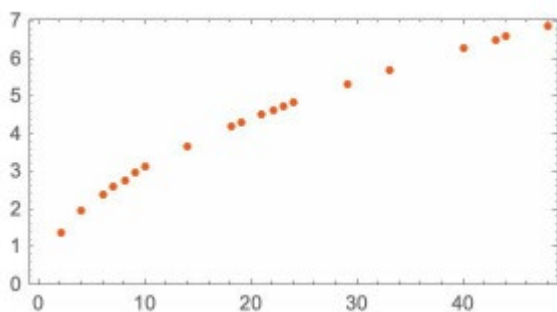
在网络抓取中，可能有几百亿个单词；在已数字化的书籍中，可能还有另外一百亿个单词。但是，在 40,000 个常用单词中，甚至可能的双字母组合就已经有 16 亿个，而三字母组合的可能性则达到了 60 万亿。因此，我们无法从已有的文本中估算出所有这些可能性的概率。而当我们到达 20 个单词的“文章片段”时，可能性的数量已经大于宇宙中的粒子数量，因此从某种意义上说，它们永远不可能全部写下来。

那我们该怎么办呢？大的想法是建立一个模型，使我们能够估计序列出现的概率，即使我们从未在我们所查看的文本语料库中明确看到过这些序列。而 ChatGPT 的核心正是一个所谓的“大型语言模型”（LLM），它被构建成能够很好地估算这些概率。

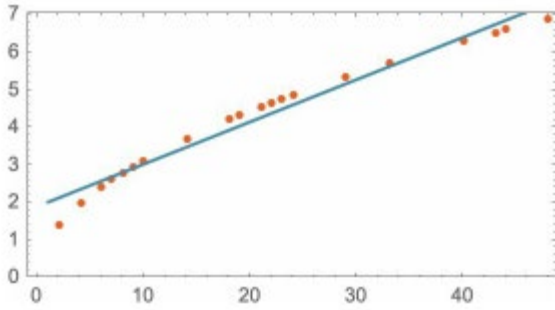
什么是模型？

比如说你想知道（就像加利略在 16 世纪后期那样）从比萨斜塔的每层楼上落下的炮弹落地需要多长时间。好吧，你可以测量每种情况并制作一个结果表。或者你可以做理论科学的本质：建立一个模型，提供一些计算答案的程序，而不是只测量并记住每种情况。

让我们假设我们有（有些理想化的）数据，记录了炮弹从各个楼层落地所需的时间：

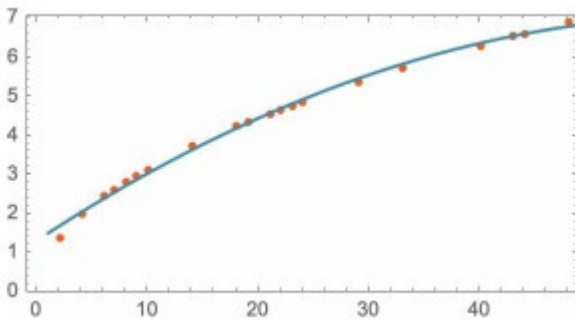


我们如何计算我们没有数据的楼层的落地时间？在这种情况下，我们可以使用已知的物理定律来计算。但是，如果我们只有数据，而不知道支配它的基本定律，我们可能会进行数学猜测，比如我们可能应该使用一条直线作为模型：

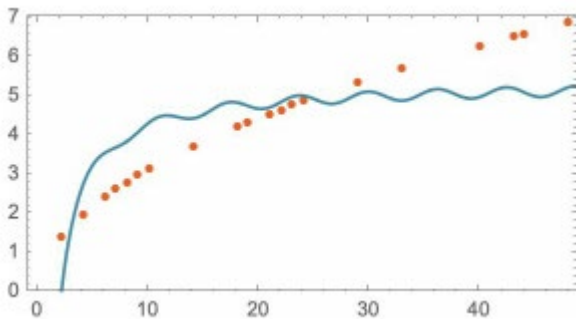


我们可以选择不同的直线。但是这条直线平均上最接近我们所给出的数据。从这条直线，我们可以估计出任何一层楼的落地时间。

我们如何知道在这里尝试使用直线？从某种程度上说，我们不知道。这只是一些数学上简单的东西，我们习惯于很多我们测量的数据与数学上简单的东西相吻合。我们可以尝试更复杂的数学方式，比如 $a + bx + cx^2$ ，在这种情况下，我们做得更好：



然而事情并不总是一帆风顺的。比如，下面这个式子是我们用 $a + b/x + c \sin(x)$ 最好的拟合：



值得理解的是，永远不存在“无模型的模型”。任何模型都有某种特定的基本结构，然后有一定数量的“旋钮”可以调整（也就是可以设置的参数）来拟合数据。在 ChatGPT 的情况下，使用了大量的这种“旋钮”，实际上有 1750 亿个。

但是令人惊奇的是，ChatGPT 的基本结构，即拥有“仅有”的那么多参数，足以建立一个计算下一个单词概率“足够好”的模型，从而生成出合理长度的文章。

人类式任务的模型

我们上面的例子涉及到制定数字数据模型，这些数据基本来自于简单物理学——几个世纪以来我们一直知道“简单的数学方法适用于这种情况”。但对于 ChatGPT，我们必须建立一个人类大脑产生的语言文本模型。而对于这样的任务，我们目前没有任何类似于“简单数学”的东西。那么它的模型可能是什么样子的呢？

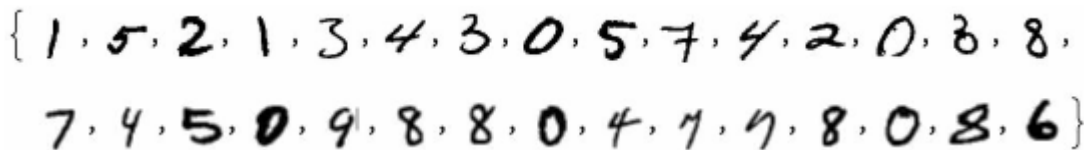
在谈论语言之前，让我们来谈论另一种人类式任务：识别图像。作为一个简单的例子，让我们考虑数字的图像（是的，这是一个经典的机器学习例子）：



我们可以做的一件事是为每个数字获取一堆样本图像：



然后，为了确定我们输入的图像是否对应于特定的数字，我们可以将其与我们拥有的样本进行明确的逐像素比较。但是作为人类，我们似乎做得更好，因为即使是手写的数字，也具有各种修改和扭曲，我们仍然能够识别出它们：



当我们为上面的数字数据制定模型时，我们能够获取我们收到的数值 x ，然后为特定的 a 和 b 计算 $a + bx$ 。因此，如果我们将每个像素的灰度值视为某个变量 x_i ，那么是否有某个函数可以对所有这些变量进行评估，告诉我们图像的数字是多少？事实证明，可以构建这样的函数。毫不奇怪的是，它并不特别简单。一个典型的例子可能涉及大约五十万个数学运算。

但最终的结果是，如果我们将图像的像素值集合输入到该函数中，它将输出指定该图像所属数字的数字。稍后，我们将谈论如何构建这样的函数以及神经网络的概念。但现在让我们将函数视为黑盒子，我们将手写数字的图像（作为像素值的数组）输入，并获得相应数字的输出：

```
In[ ]:= NetModel [{"..." +}] [{"7, 0, 9, 7, 8, 2, 4, 1, 1, 1}]
```

```
Out[ ]:= {7, 0, 9, 7, 8, 2, 4, 1, 1, 1}
```

但实际上在这里发生了什么？假设我们逐渐模糊一位数字。一段时间内，我们的函数仍然“识别”它，这里是一个“2”。但很快，它“失去了”，开始给出“错误”的结果：

```
In[ ]:= NetModel [{"..." +}] [{"2, 2, 2, 2, 2, 2, 2, 2, 2, 2}]
```

```
Out[ ]:= {2, 2, 2, 1, 1, 1, 1, 1, 1, 1}
```

但是为什么我们说这是“错误”的结果呢？在这种情况下，我们知道我们通过模糊“2”来获得了所有图像。但是，如果我们的目标是产生一个关于人类识别图像模型，真正需要问的问题是，如果呈现其中一个模糊的图像，而不知道它来自哪里，人类会怎么做。

如果我们的函数得到的结果通常与人类的判断相一致，那么我们就有了一个“好的模型”。而这个非平凡的科学事实是，对于像图像识别这样的任务，我们现在基本上知道如何构建这样的函数。

我们能否“数学证明”它们有效？嗯，不能。因为要做到这一点，我们必须拥有一个关于人类所做的事情的数学理论。拿“2”图像并改变一些像素。我们可以想象，即使只有几个像素“错位”，我们仍应该认为这是一个“2”图像。但这应该持续多久呢？这是一个关于人类视觉感知的问题。是的，答案毫无疑问对于蜜蜂或章鱼是不同的——对于假想的外星人可能完全不同。

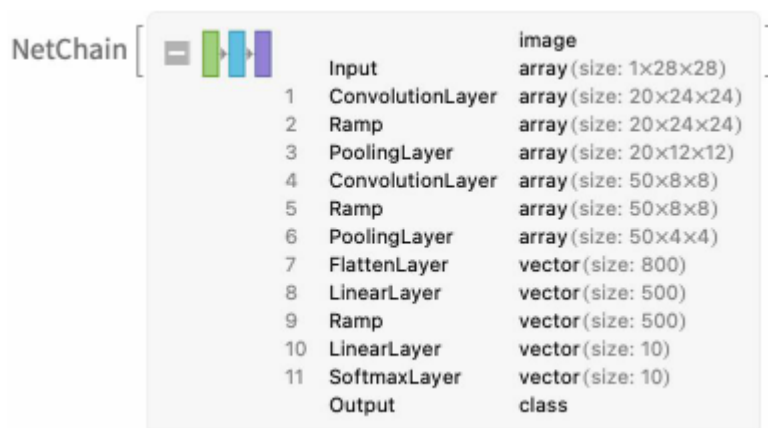
神经网络

好的，那么我们通常用于图像识别等任务的模型实际上是如何工作的呢？目前最流行且最成功的方法是使用神经网络。神经网络是在 1940 年代发明的，其形式与今天使用的形式非常接近，可以被视为大脑工作方式的简单理想化。

人类大脑中有约 1000 亿个神经元（神经细胞），每个神经元可以产生电脉冲，每秒最多约千次。神经元以复杂的方式相连，每个神经元都有树状分支，使其可以向可能有数千个神经元传递电信号。在粗略的近似下，任何给定神经元是否在给定时刻产生电脉冲取决于其从其他神经元接收到的脉冲，不同的连接以不同的“权重”贡献。

当我们“看到一幅图像”时，从图像上的光子落在我们眼睛后面的（光感受器）细胞上时，它们会在神经元中产生电信号。这些神经元与其他神经元相连，并最终通过一系列神经元层传递信号。在这个过程中，我们“识别”图像，并最终“形成想法”，即“看到数字 2”（最后可能会做一些像大声说“二”之类的事情）。

上一节中的“黑盒子”函数是这种神经网络的“数学化”版本。它恰好有 11 个层（尽管只有 4 个“核心层”）：



这种神经网络没有特别“理论派生”的地方，它只是在 1998 年作为一种工程构造出来的，并被发现可行。（当然，这与我们可能会描述我们的大脑是通过生物进化的过程产生的方式并没有太大的区别。）

好的，但是像这样的神经网络是如何“识别东西”的呢？关键在于吸引子的概念。假设我们有手写的数字 1 和 2 的图像：



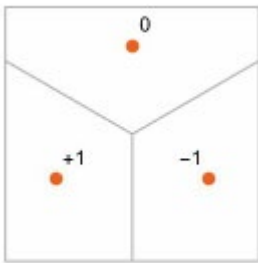
我们希望所有数字 1“被吸引到一个地方”，所有数字 2“被吸引到另一个地方”。或者换一种说法，如果一个图像与数字 1 相比“更接近”，那么我们希望它最终出现在“数字 1 的位置”，反之亦然。

作为一个直观比喻，假设我们有平面上的某些位置，用点表示（在现实场景中可能是咖啡店的位置）。然后我们可以想象，从平面上任何一个点开始，我们总是想要最终到达最近的点（即我们总是去最近的咖啡店）。我们可以通过将平面分成由理想化的“分水岭”分开的区域（“吸引子盆地”）来表示这一点：

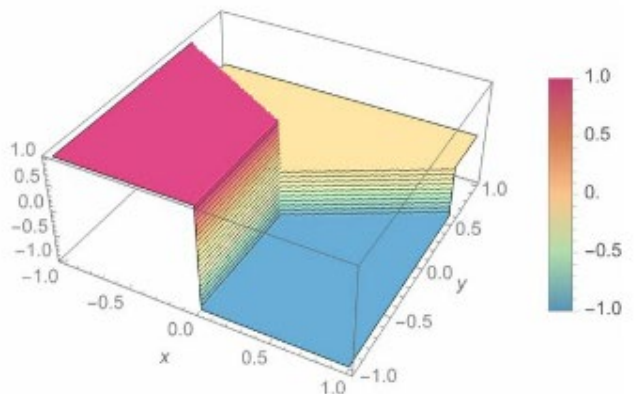
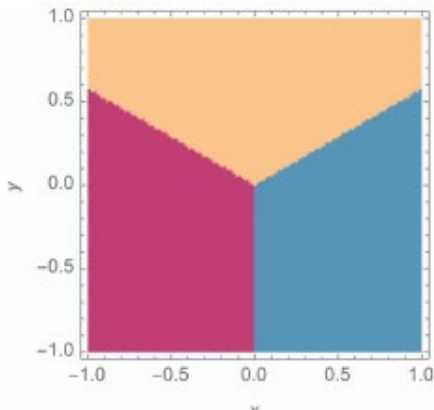


我们可以将其看作是实现一种“识别任务”，我们不是在做像识别给定图像“最像哪个数字”这样的事情，而是直接看哪个点距离给定点最近。（我们在这里展示的“Voronoi 图”设置是将二维欧几里德空间中的点分开；数字识别任务可以被认为是在由每个图像中所有像素的灰度级形成的 784 维空间中执行非常类似的操作。）

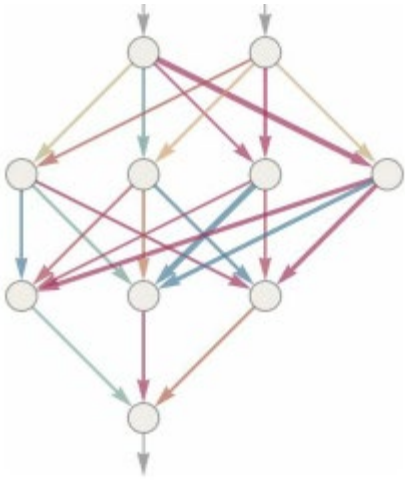
那么，我们如何让神经网络“执行识别任务”呢？让我们考虑这个非常简单的情况：



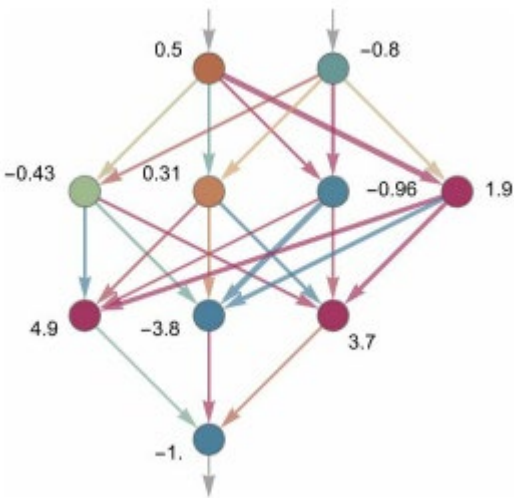
我们的目标是将一个与位置 $\{x, y\}$ 相对应的“输入”识别为最接近的三个点之一。或者换句话说，我们希望神经网络计算出 $\{x, y\}$ 的函数，如下所示：



那么如何用神经网络实现呢？最终，神经网络是一个连接的理想化“神经元”集合，通常按层排列，一个简单的例子如下：



每个“神经元”实际上是设置为评估简单数值函数。要“使用”网络，我们只需在顶部输入数字（例如我们的坐标 x 和 y ），然后让每个层上的神经元“评估其函数”并通过网络向前传递结果，最终在底部产生最终结果：



传统的（仿生学启发的）设置中，每个神经元实际上有来自上一层神经元的一定数量的“输入连接”，每个连接被分配一个特定的“权重”（可以是正数或负数）。给定神经元的值是通过将“前置神经元”的值乘以其相应的权重，然后加起来并加上常数，最后应用“阈值”（或“激活”）函数来确定的。在数学上，如果一个神经元有输入 $x = \{x_1, x_2 \dots\}$ ，那么我们计算 $f[w \cdot x + b]$ ，其中权重 w 和常数 b 通常为网络中的每个神经元选择不同的值；函数 f 通常是相同的。

计算 $w \cdot x + b$ 只是矩阵乘法和加法的问题。“激活函数” f 引入了非线性（最终是导致非平凡行为的原因）。通常使用各种激活函数；这里我们将使用 Ramp（或 ReLU）：



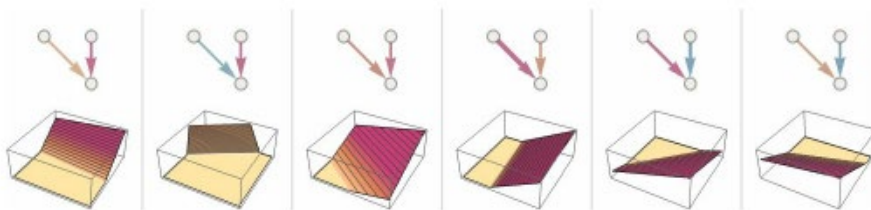
对于我们希望神经网络执行的每个任务（或等效地，对于我们希望其评估的每个整体函数），我们都将有不同的权重选择。（正如我们将在稍后讨论的那样，这些权重通常是通过使用机器学习从我们想要的输出示例来“训练”神经网络确定的。）

最终，每个神经网络都只对应于某个整体数学函数-尽管可能难以写出来。对于上面的例子，它将是：

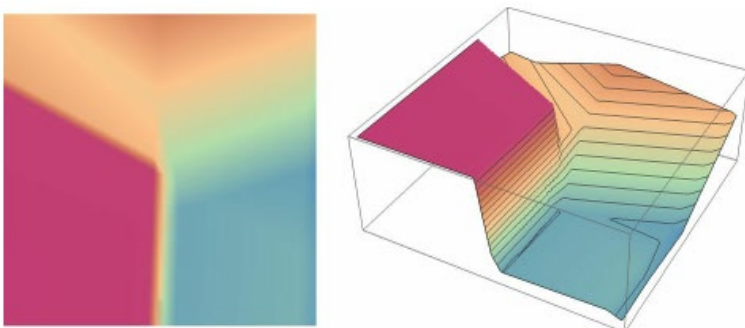
$$w_{511}f(w_{311}f(b_{11} + xw_{111} + yw_{112}) + w_{312}f(b_{12} + xw_{121} + yw_{122}) + w_{313}f(b_{13} + xw_{131} + yw_{132}) + w_{314}f(b_{14} + xw_{141} + yw_{142}) + b_{31}) + w_{512}f(w_{321}f(b_{11} + xw_{111} + yw_{112}) + w_{322}f(b_{12} + xw_{121} + yw_{122}) + w_{323}f(b_{13} + xw_{131} + yw_{132}) + w_{324}f(b_{14} + xw_{141} + yw_{142}) + b_{32}) + w_{513}f(w_{331}f(b_{11} + xw_{111} + yw_{112}) + w_{332}f(b_{12} + xw_{121} + yw_{122}) + w_{333}f(b_{13} + xw_{131} + yw_{132}) + w_{334}f(b_{14} + xw_{141} + yw_{142}) + b_{33}) + b_{51}$$

ChatGPT 的神经网络也只对应于像这样的数学函数-但实际上有数十亿个项。

但是让我们回到单个神经元。这里是一个带有两个输入（表示坐标 x 和 y ）的神经元在使用各种不同的权重和常数（并使用 Ramp 作为激活函数）时可以计算的一些示例函数：

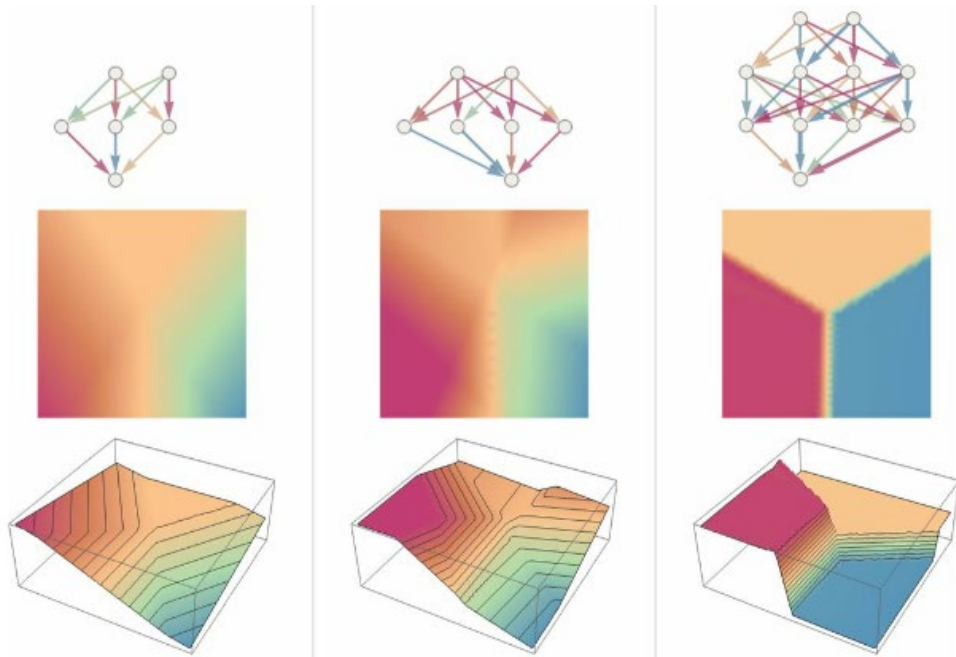


但是上面的大型神经网络呢？嗯，这是它的计算结果：



它并不完全“正确”，但它接近我们上面展示的“最近点”函数。

让我们看看其他一些神经网络的计算结果。在每种情况下，正如我们稍后将解释的那样，我们使用机器学习来找到最佳的权重选择。然后在这里显示该权重的神经网络计算结果：



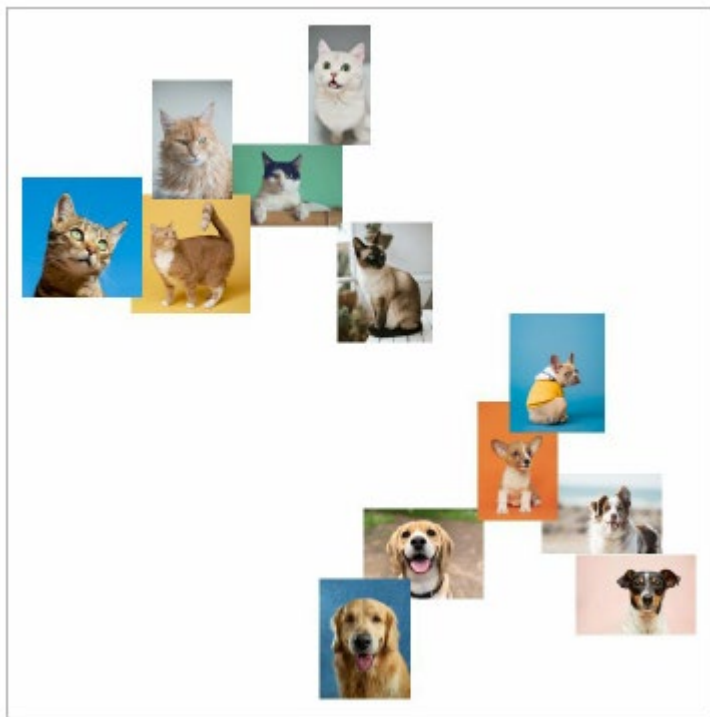
更大的神经网络通常更能逼近我们所追求的函数。在每个吸引子盆地的中心，通常我们可以得到我们想要的确切答案。但在边界处——神经网络“犹豫不决”的地方——事情可能会更混乱。

在这种简单的数学式“识别任务”中，正确答案很明显。但在识别手写数字的问题中，情况就不那么清楚了。如果有人把“2”写得像“7”怎么办？尽管如此，我们可以问一下神经网络是如何区分数字的——这就是一个线索：



我们能否“数学地”说明神经网络如何进行区分？其实不能。它只是“在做神经网络该做的事情”。但事实证明，这通常与我们人类所做的区分相当一致。

让我们举一个更详细的例子。假设我们有猫和狗的图像。我们有一个已经被训练出来区分它们的神经网络。以下是它在一些例子上的表现：



现在，我们不太清楚“正确答案”是什么。如果有只穿着猫装的狗呢？等等。无论输入是什么，神经网络都会生成一个答案。而且，事实证明，它以一种与人类相当一致的方式进行操作。正如我之前所说的，这不是我们可以“从第一原理”推导出的事实。这只是在某些领域中已被证明为真的一个关键原因。但这也是神经网络有用的一个关键原因：它们以某种方式捕捉到了“类似于人类”的做事方式。

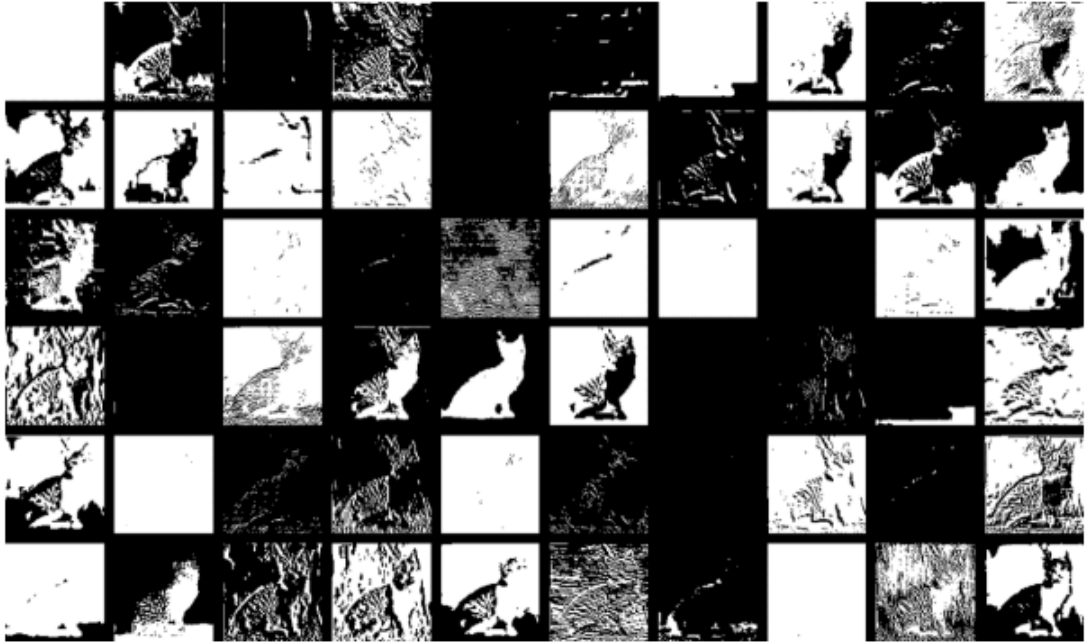
给自己看一张猫的图片，然后问“为什么这是只猫？”。也许你会开始说“嗯，我看到了它的尖耳朵等等”。但很难解释你是如何识别出这张猫的图片的。这只是因为你的大脑以某种方式找到了答案。但对于大脑来说，目前没有办法“进入内部”并看到它是如何找到答案的。对于（人造）神经网络呢？当你展示一张猫的图片时，很容易看到每个“神经元”做了什么。但即使是得到一个基本的可视化通常也非常困难。

在上面用于“最近点”问题的最终网络中有 17 个神经元，在用于识别手写数字的网络中有 2190 个神经元，在用于识别猫和狗的网络中有 60,650 个神经元。通常情况下，将 60,650 维空间可视化是相当困难的。但是，由于这是一个处理图像的网络，因此它的许多神经元层被组织成数组，就像它正在查看的像素数组一样。

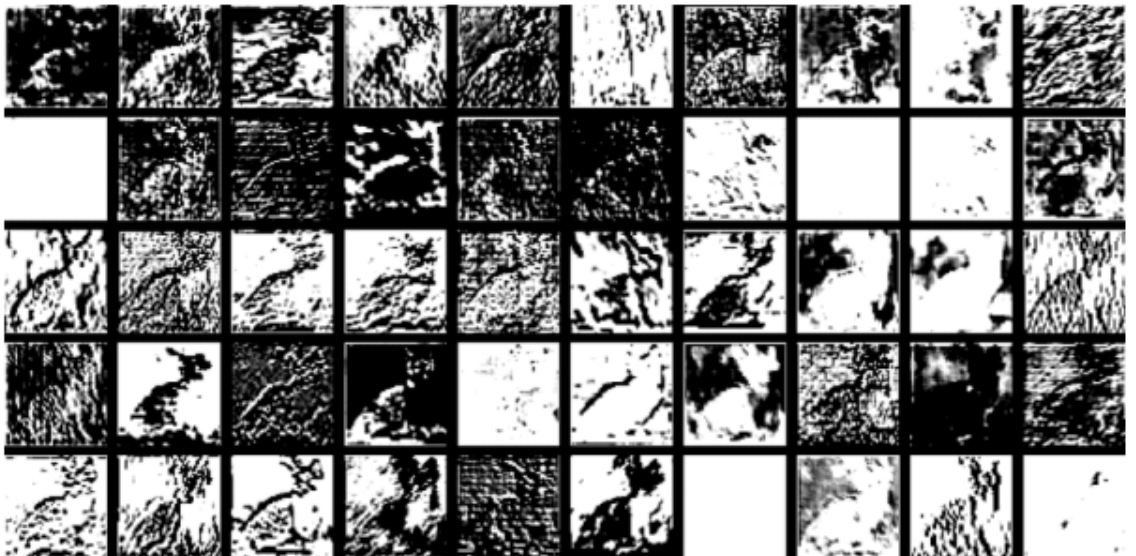
如果我们拿一个典型的猫图像：



那么我们可以通过一系列派生图像来表示第一层神经元的状态，其中许多图像我们可以轻松地解释为“没有背景的猫”，或者“猫的轮廓”：



到了第 10 层，很难解释正在发生什么：



但总的来说，我们可以说神经网络正在“挑选某些特征”（也许尖耳朵是其中之一），并使用这些特征确定图像的内容是什么。但是，这些特征是否是我们可以命名的特征，例如“尖耳朵”？大多数不是。

我们的大脑是否也使用类似的特征？大多数情况下我们不知道。但值得注意的是，像这里展示的神经网络的前几层似乎挑选出了图像的某些方面（例如物体的边缘），这些方面似乎类似于我们知道在大脑的视觉处理的第一层中被挑选出来的方面。

但是假设我们想要一个神经网络的“猫识别理论”。我们可以说：“看，这个特定的网络可以做到”——这立即给我们一些“问题的难度”的概念（例如需要多少神经元或层）。但至少目前为止，我们没有一种“提供一个叙述性描述”的方法来说明网络正在做什么。也许这是因为它真的是计算上不可约的，除了显式跟踪每一步，我们没有一般的方法来找到它所做的事情。或者可能只是因为我们还没有“弄清楚科学”，并确定了“自然法则”，使我们能够总结正在发生的事情。

当我们谈论使用 ChatGPT 生成语言时，我们将遇到相同的问题。同样，不清楚是否有方法“总结它正在做什么”。但是，语言的丰富性和细节（以及我们对它的经验）可能会使我们比图像更进一步。

机器学习和神经网络的训练

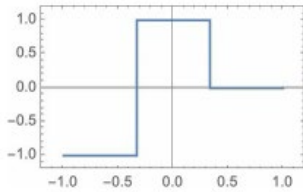
到目前为止，我们已经讨论了“已经知道”如何执行特定任务的神经网络。但神经网络之所以如此有用（假定在大脑中也是如此），是因为它们不仅原则上可以执行各种任务，而且可以通过逐步“从示例中训练”来执行这些任务。

当我们制作一个神经网络来区分猫和狗时，我们不需要有效地编写一个程序来（例如）明确地找到胡须。相反，我们只需展示大量示例，说明什么是猫，什么是狗，然后让网络从中“机器学习”如何区分它们。

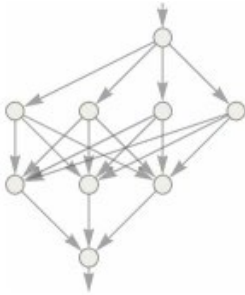
关键在于训练后的网络从特定的示例中“概括出”通用的规律。就像我们上面看到的那样，网络并不仅仅是识别它所展示的某个猫图像的特定像素模式；相反，神经网络以某种“一般猫”的特征来区分图像。

那么神经网络的训练是如何工作的呢？基本上，我们总是在尝试找到权重，使得神经网络能够成功地再现我们所给的示例。然后，我们依赖神经网络在一种“合理”的方式下“插值”（或“概括”）这些示例。

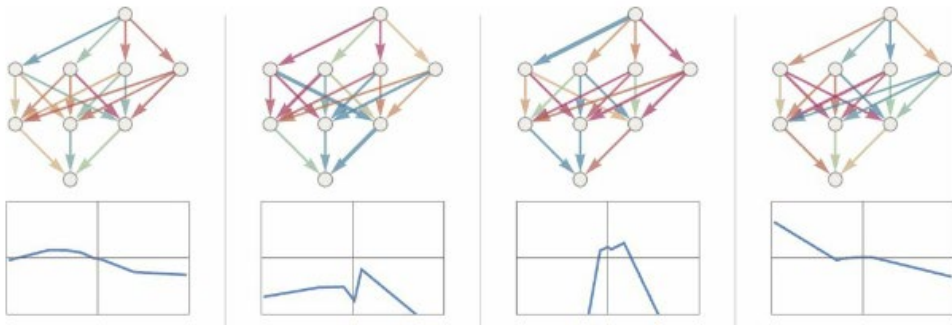
让我们来看一个比上面最近点问题更简单的问题。我们只是试图让神经网络学习函数：



对于这个任务，我们需要一个只有一个输入和一个输出的网络，如下所示：

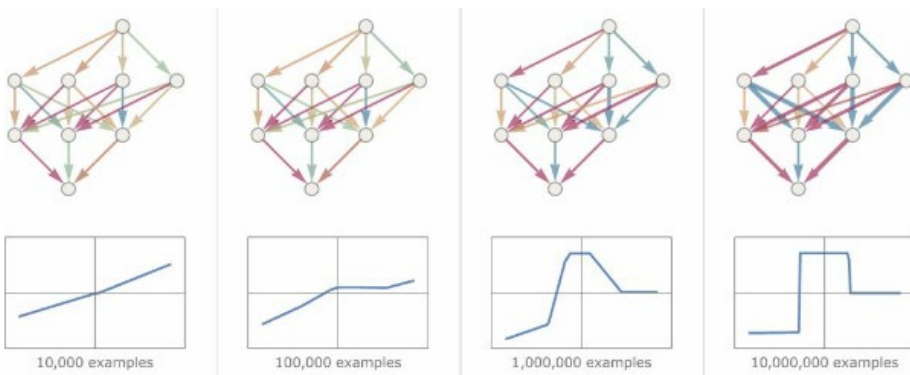


但是我们应该使用什么权重呢？每组可能的权重都会计算出一些函数。例如，以下是使用几组随机选择的权重所得到的结果：



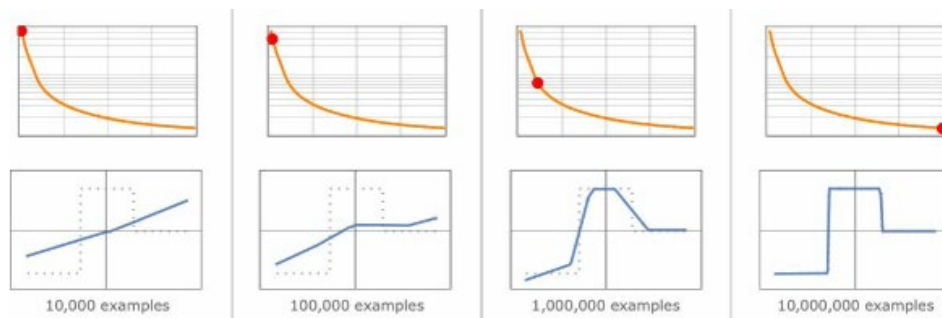
是的，我们可以明显看到，在这些情况下，它都没有成功地接近我们想要的函数。那么我们如何找到能够复现该函数的权重呢？

基本思路是提供大量的“输入→输出”示例进行“学习”，然后尝试找到能够复现这些示例的权重。以下是逐步增加示例数量的结果：



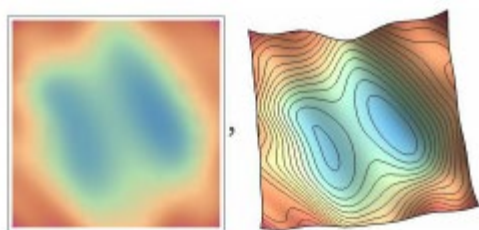
在每个“训练”阶段，网络中的权重都在逐步调整，我们可以看到最终得到了一个成功复制所需函数的网络。那么我们如何调整权重呢？基本思路是在每个阶段看“我们与目标的距离有多远”，然后更新权重以使之更接近。

为了了解“我们与目标的距离有多远”，我们计算通常称为“损失函数”（有时称为“代价函数”）。在这里，我们使用简单的（L2）损失函数，它只是获取值和真实值之间差值的平方和。我们可以看到，随着训练过程的进行，损失函数逐步减小（遵循不同任务的不同“学习曲线”）-直到达到一个点，网络（至少可以很好地近似）成功地复制所需的函数：

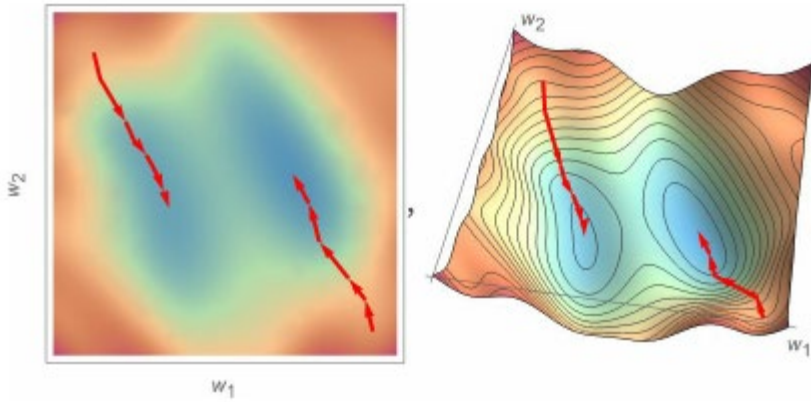


好的，最后一个必要的部分需要解释的是如何调整权重来减少损失函数。正如我们所说，损失函数给出了我们得到的值和真实值之间的“距离”。但是“我们得到的值”是由当前版本的神经网络确定的，并且由其中的权重确定。但是现在想象一下，权重是变量，比如 w_i 。我们想知道如何调整这些变量的值以最小化取决于它们的损失。

例如，假设（极大地简化了实际使用的典型神经网络），我们只有两个权重 w_1 和 w_2 。那么我们可能会有一个损失，它关于 w_1 和 w_2 的函数看起来像这样：



数值分析提供了各种技术，用于在这种情况下找到最小值。但是典型的方法是从任何以前的 w_1 , w_2 开始逐步遵循最陡峭的下降路径：



就像水从山上流下来一样，这个过程只能保证最终会到达表面的某个局部最小值（“山脊湖”）；它可能不会达到最终的全局最小值。

不明显的是，在“重量景观”上找到最陡峭的下降路径是否可行。但是微积分来拯救我们。正如我们上面提到的，我们始终可以将神经网络视为计算一种数学函数-这种函数取决于它的输入和权重。但是现在考虑相对于这些权重微分。结果表明，微积分的链式法则实际上让我们“解开”了神经网络中的连续层所做的操作。结果是我们可以在某种局部逼近中“反演”神经网络的操作，并逐步找到最小化与输出相关联的损失的权重。

上面的图片展示了我们在只有 2 个权重的情况下可能需要做的最小化。但事实证明，即使有更多的权重（ChatGPT 使用了 1750 亿个），仍然有可能做到最小化，至少在某种程度上的逼近。实际上，“深度学习”在 2011 年左右的重大突破与发现有关，即在某种意义上，当涉及许多权重时，与涉及相当少权重时相比，最小化可能更容易。

换句话说，相对来说，使用神经网络解决更复杂的问题可能比解决更简单的问题更容易，这似乎有些反直觉。粗略的原因似乎是，当有很多“权重变量”时，就会有一个高维空间，其中“有很多不同的方向”可以引导我们到达最小值，而当变量较少时，更容易陷入局部最小值（“山湖”），没有“方向可以走出去”。

值得指出的是，在典型情况下，有许多不同的权重集合将都可以产生基本相同的性能的神经网络。通常在实际的神经网络训练中会进行许多随机选择，这会导致“不同但等效的解决方案”，例如这些：



但每个这样的“不同解决方案”都将具有至少稍微不同的行为。如果我们要求在我们提供的训练示例区域之外进行“外推”，我们可能会得到截然不同的结果。



但哪一个“正确的”呢？真的没有办法说。它们都“与观察到的数据一致”。但它们都对应着不同的“天生”的思考方式，即如何“考虑”如何在“盒子外部”执行任务。一些可能比其他人对我们人类更“合理”。

神经网络训练的实践和经验

特别是在过去的十年里，神经网络的训练艺术取得了许多进展。是的，这基本上是一门艺术。有时候，特别是事后回顾，我们能看到某些被采取的做法至少有点科学解释。但是大多数情况下，这些都是通过试错发现的，逐步添加了一些想法和技巧，积累了相当丰富的神经网络操作经验。

其中有几个关键点。首先，对于特定任务，应该使用什么样的神经网络结构。然后是如何获得神经网络的训练数据的关键问题。并且越来越多地不是从头训练网络：新的网络可以直接集成另一个已经训练好的网络，或者至少可以使用该网络生成更多的训练示例。

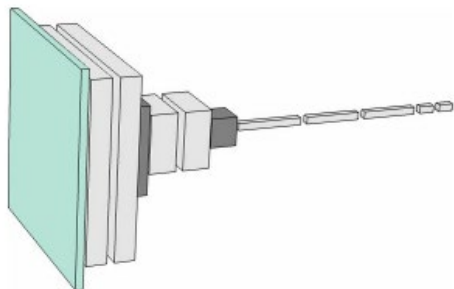
人们可能认为，对于每种特定的任务，都需要不同的神经网络结构。但是所发现的是，相同的结构通常似乎即使对于明显不同的任务也可以工作。在某种程度上，这让人想起了通用计算的概念（以及我关于计算等价性原理的观点），但正如我稍后将讨论的那样，我认为这更反映了我们通常试图让神经网络执行的任务是“类似人类”的，而神经网络可以捕捉相当普遍的“类似人类的过程”。

在神经网络的早期阶段，人们倾向于认为应该“尽可能让神经网络少做事情”。例如，在将语音转换为文本时，认为应该先分析语音，将其分解为音素等。但是发现，对于“类似人类的任务”，最好的方法通常是让神经网络尝试从端到端解决问题，让它“发现”必要的中间特征、编码等。

还有一个想法是将复杂的独立组件引入神经网络中，以让它们实际上“显式实现特定的算法思想”。但同样地，这在大多数情况下都被证明不值得；相反，更好的方法是使用非常简单的组件，并让它们“自我组织”（虽然通常是以我们无法理解的方式）来实现（可能是）等价的算法思想。

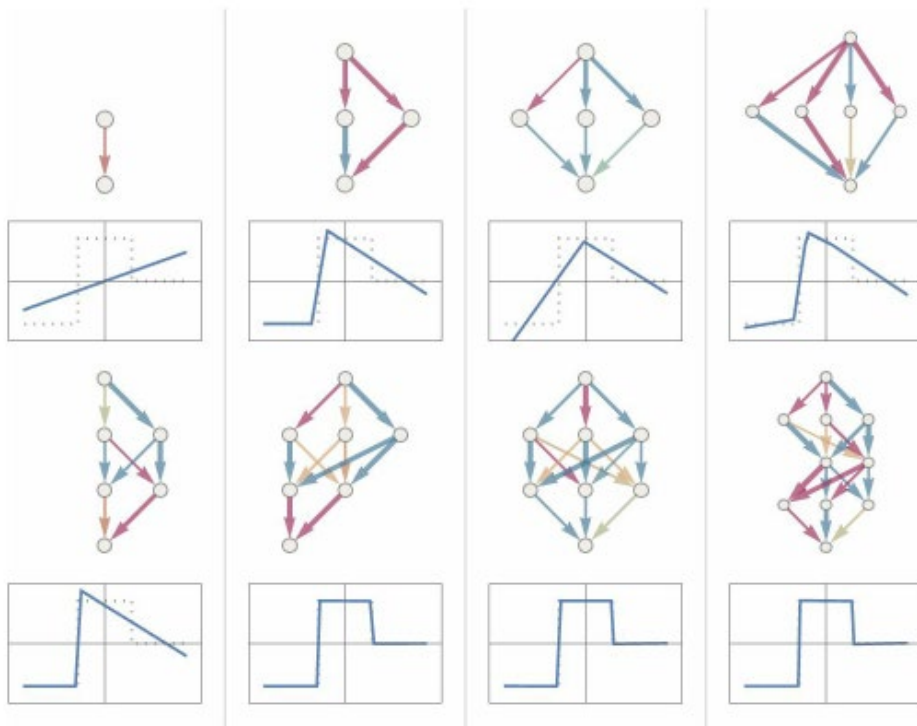
这并不意味着没有与神经网络相关的“结构性想法”。例如，使用具有局部连接的 2D 神经元阵列在处理图像的早期阶段似乎至少非常有用。而具有集中于“向后查看序列”的连接模式似乎有用——正如我们后面将在 ChatGPT 中看到的那样——处理诸如人类语言之类的事物。

但神经网络的一个重要特点是——就像计算机一样——它们最终只是处理数据。而当前的神经网络——与当前的神经网络训练方法一样——特别处理数字数组。但在处理过程中，这些数组可以完全重新排列和重塑。作为一个例子，我们上面用于识别数字的网络从一个 2D“图像样式”的数组开始，很快“变厚”到许多通道，然后“集中”到一个 1D 数组中，最终包含代表不同可能输出数字的元素：



但是，好吧，怎么才能确定一个特定任务所需要的神经网络有多大呢？这是一种艺术。在某种程度上，关键是要知道“任务有多难”。但是对于像人类一样的任务，通常很难估计。是的，可能有一种机械的方式来做这个任务。但是很难知道是否有所谓的技巧或快捷方式，允许我们以“人类水平”更轻松的方式完成任务。可能需要枚举一个巨大的游戏树才能“机械化”地玩某个游戏；但是可能有一种更容易的（“启发式”）方法来实现“人类级别的游戏”。

当处理微小的神经网络和简单的任务时，有时可以明确看到从这里“无法到达那里”。例如，这是使用一些小型神经网络在上一节任务上的最佳表现：



我们可以看到，如果网络太小，它就无法重现我们想要的函数。但是在一定的大小以上，它没有问题——至少如果我们为其提供足够的训练样本，进行足够长的训练。顺便说一下，这些图片说明了神经网络的一个技巧：如果在中间有“挤压”，迫使所有东西通过一个更小的中间神经元数量，通常可以用更小的网络。（值得一提的是，“无中间层”或所谓的“感知器”网络只能学习基本上是线性的函数——但是只要有一个中间层，就总体上可以在原则上以任意高的精度逼近任何函数，至少如果有足够的神经元，但为了使其可行地训练，通常具有某种正则化或归一化。）

好的，假设我们已经确定了特定的神经网络架构。现在问题在于如何获取用于训练网络的数据。对于神经网络和机器学习的许多实际挑战都集中在获取或准备必要的训练数据上。在许多情况下（“监督学习”），人们希望获得输入的明确示例以及预期的输出。例如，人们可能希望标记图像中包含的内容或其他属性。也许人们需要明确地进行标记，通常需要大量的工作。但很多时候，人们发现可以借助已经完成的工作或使用它作为某种代理。例如，人们可能会使用已经提供在网络上图像的 alt 标签。或者，在另一个领域中，人们可以使用为视频创建的闭路字幕。或者，在语言翻译培训方面，人们可以使用存在于不同语言中的网页或其他文档的平行版本。

您需要向神经网络展示多少数据才能训练它完成特定任务？同样，很难从原则上进行估计。当然，使用“转移学习”将已在另一个网络中学习到的重要功能列表“转移”到网络中可以大大减少要求。但通常，神经网络需要“看到很多示例”才能进行良好的训练。对于某些任务而言，每个示例可以具有极高的重复性，这是神经网络传统的重要部分。实际上，标准策略是将所有示例重复显示给神经网络。在这些“训练轮次”（或“周期”）中，神经网络将处于至少稍微不同的状态，并且以某种方式“提醒它”特定示例对于帮助它“记住该示例”是有用的。（是的，这也许类似于人类记忆中重复的有用性。）

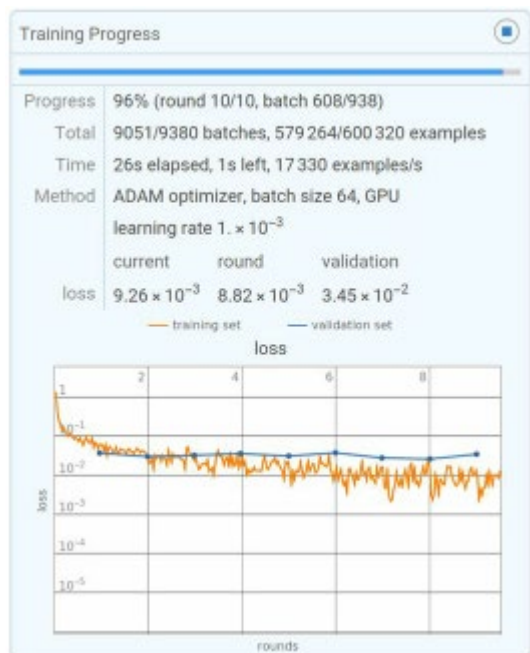
但是，仅重复相同的示例并不足够。还需要向神经网络展示示例的变化。神经网络传统的特征是，这些“数据增强”变化并不需要很复杂就可以很有用。仅使用基本图像处理略微修改图像就可以使其在神经网络训练中“崭新如初”。同样，当实际视频数据用尽时，人们可以继续类似于模型视频游戏的环境中运行模拟并获取数据，而无需所有真实世界场景的细节，这也是自动驾驶汽车训练的一种标准策略。

那么，对于像 ChatGPT 这样的模型呢？它有一个很好的特点，即可以进行“无监督学习”，使得从中获得示例进行训练变得更加容易。回想一下，ChatGPT 的基本任务是找出如何继续给定的文本。因此，要获得“训练示例”，我们只需要获得一段文本，将其末尾遮蔽，然后将其用作“训练输入”——而“输出”则是完整的、未被遮蔽的文本。我们稍后会详细讨论这个问题，但主要的一点是，与例如学习图像中的内容的情况不同，这里不需要进行“明确标记”；ChatGPT 实际上可以直接从任何文本示例中学习。

那么，在神经网络中的实际学习过程是怎样的呢？最终，它都是确定哪些权重最能捕捉到所给出的训练示例。有各种各样的详细选择和“超参数设置”（所谓的权重可以被认为是“参数”），可用于调整如何进行这样的权重调整。有不同的损失函数选择（平方和、绝对值和等）。有不同的损失最小化方法（每一步在权重空间移动的距离等）。然后，还

有一些问题，例如每次要显示多少个示例以获取每个连续的损失估计。是的，我们可以应用机器学习（例如在 Wolfram 语言中）来自动化机器学习，以及自动设置像超参数这样的事。

但最终，整个训练过程可以通过观察损失如何逐步减小来描述（例如，这是一个小型训练的 Wolfram 语言进度监视器）：



那么，对于类似 ChatGPT 这样的任务怎么办呢？嗯，它有一个很好的特点，就是它可以进行“无监督学习”，这样就更容易得到用于训练的示例。回想一下，ChatGPT 的基本任务是找出如何继续给定的一段文本。因此，要获得“训练示例”，我们只需要获取一段文本，并将其末尾的部分遮蔽，然后将其用作“训练输入”，“输出”是完整的、未遮蔽的文本。我们稍后会讨论这个问题，但主要的观点是，与学习图像内容等任务不同，这里不需要“显式标记”，ChatGPT 实际上可以直接从给定的任何文本示例中学习。

那么，神经网络中的实际学习过程是什么样的呢？最终，它都是关于确定哪些权重最能捕捉到已经给定的训练示例的过程。有各种各样的详细选择和“超参数设置”（所谓的“参数”是指权重），可以用来调整这个过程。有不同的损失函数选择（平方和、绝对值之和等）。有不同的损失最小化方法（每一步在权重空间中移动多远等）。然后有像“批次”大小这样的问题，即每次展示多少个示例来获取正在尝试最小化的损失的每个连续估计值。是的，可以应用机器学习（就像我们在 Wolfram Language 中所做的那样），来自动化机器学习，并自动设置像超参数之类的东西。

但最终，整个训练过程可以通过观察损失如何逐步减少来描述（例如这个小训练的 Wolfram Language 进度监视器）：

在像细胞自动机这样的计算系统中，基本上是并行处理许多个体比特，从而进行这种增量修改从未有过清晰的方法，但没有理由认为这是不可能的。实际上，就像“2012 年深度学习突破”一样，这种增量修改在更复杂的情况下可能会更容易实现。

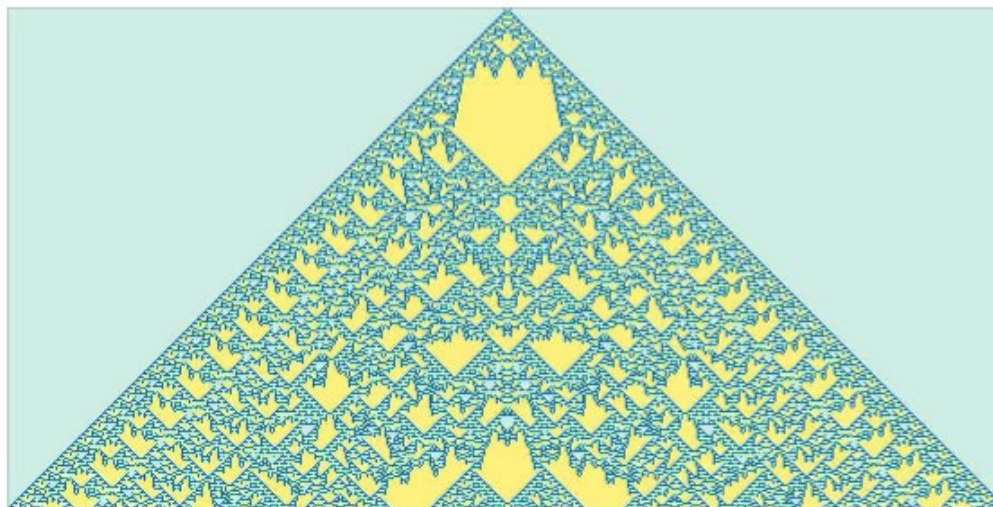
神经网络——也许有点像大脑——被设置为具有基本固定的神经元网络，其中修改的是它们之间连接的强度（“权重”）。（也许在至少是年轻的大脑中，全新的连接数量也会显著增加。）但是，虽然这可能是生物学中方便的设置，但它根本不清楚它是否接近于实现我们所需的功能的最佳方式。可能涉及相当于渐进网络重写的东西（也许类似于我们的物理项目），最终可能更好。

但即使在现有神经网络的框架内，目前存在着关键限制：现在所做的神经网络训练基本上是顺序的，每批样例的影响会传播回来更新权重。实际上，即使考虑到 GPU，当前计算机硬件的大部分时间在训练期间都是“空闲的”，只有一部分在更新。从某种意义上说，这是因为我们当前的计算机倾向于具有与其 CPU（或 GPU）分离的内存。但是在大脑中，这可能是不同的——每个“内存元素”（即神经元）也是一个潜在的主动计算元素。如果我们能够以这种方式设置未来的计算机硬件，可能会更有效地进行训练。

“当然，只要神经网络足够大，就能做任何事情！”

像 ChatGPT 这样的东西的能力看起来如此令人印象深刻，以至于人们可能会想象，如果只是“继续进行”，并训练越来越大的神经网络，那么它们最终就能“做任何事情”。如果人们关心的是能够立即人类思考的事情，那么这很可能是正确的。但是，过去几百年科学的教训是，有些可以通过形式化过程来解决的事情，但不容易立即接近人类思维。

非平凡的数学就是一个很好的例子。但是一般情况是计算。最终问题是计算不可化简的现象。有些计算可能需要多个步骤才能完成，但实际上可以“化简”为相当直接的东西。但是计算不可化简的发现表明，这并不总是有效的。相反，存在一些过程——可能像下面这样——必须本质上跟踪每个计算步骤才能工作：



我们通常用大脑做的事情很可能是特别选择避免计算不可化简的。在大脑中进行数学运算需要特殊的努力。实际上，要“思考”任何非平凡程序的操作步骤是基本上不可能的。

“当然，足够大的网络可以做任何事情！”

像 ChatGPT 这样的系统的能力非常惊人，以至于人们可能想象只要“不断扩大规模”，训练更大的神经网络，它们最终就能“做到一切”。如果我们关注那些立即可见的人类思考的事情，那么这很可能是正确的。但是过去几百年的科学教导我们，有些可以通过形式过程解决的问题，但人类思考无法轻易获得。

非平凡数学就是一个很好的例子。但一般情况下是计算。最终的问题是计算不可约性的现象。有些计算可能需要很多步骤才能完成，但实际上可以“缩减”为非常直接的东西。但计算不可约性的发现意味着，这种情况并不总是适用的。相反，可能存在过程，像下面这个过程一样，无论如何都需要追踪每一步计算：

我们通常用大脑处理的东西可能是特别选择的，以避免计算不可约性。人们需要特别的努力才能在大脑中进行数学计算。实际上，在大脑中仅靠思考就可以“推断”任何非平凡程序的操作步骤是很难的。

或者说，能力和可训练性之间存在一种最终的权衡：您希望系统更多地利用其计算能力，它将会展示计算不可约性，并且它将会更难训练。而如果它基本上是可训练的，那么它将无法进行复杂的计算。（对于当前的 ChatGPT 来说，情况更为极端，因为用于生成每个输出标记的神经网络是一个纯粹的“前馈”网络，没有循环，因此无法进行任何具有非平凡“控制流”计算。）

当然，人们可能会想知道是否真的有必要能够进行不可约计算。事实上，在人类历史的大部分时间里，这并不是特别重要的。但是我们现代的技术世界是建立在利用至少数学计算（越来越多地也是更一般的计算）的工程学之上的。如果我们看看自然界，那里充满了不可约计算——我们正在慢慢地理解如何模拟和利用它们来实现我们的技术目标。

当然，神经网络肯定可以注意到自然界中我们也可能很容易注意到的那些规律。但是，如果我们想解决数学或计算科学所涉及的问题，神经网络就无法做到这一点——除非它实际上“使用作为工具”的是一个“普通”的计算系统。

但是，所有这些都可能令人感到困惑。过去有很多任务——包括写作文——我们认为计算机在某种程度上“基本上太难了”。现在我们看到像 ChatGPT 这样的任务完成后，我们倾向于认为计算机必须已经变得强大得多——尤其是超越了它们已经基本能够完成的事情（例如逐步计算细胞自动机等计算系统的行为）。

但这并不是正确的结论。计算不可约过程仍然是计算不可约的，对于计算机来说仍然是基本难度，即使计算机可以轻松计算它们的每个步骤。相反，我们应该得出的结论是，像写论文这样我们人类可以做到但我们认为计算机无法做到的任务，在某种意义上计算上比我们想象的更容易。

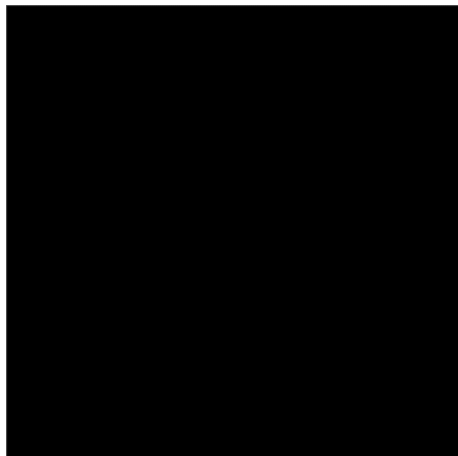
换句话说，神经网络能够成功地写一篇论文的原因是写论文原来比我们想象的“计算上更浅显易懂”。在某种意义上，这使我们更接近于“拥有一个理论”，即我们如何处理语言和写作等事情。

如果你有足够大的神经网络，那么是的，你可能能够做到人类能轻松完成的任何事情。但你无法掌握自然界普遍能做到的东西，或者我们从自然界中制造的工具所能做到的。正是这些工具的使用，包括实践和概念上的工具，使我们在最近几个世纪里超越了“纯粹依靠人类思考”所能触及的边界，捕捉了更多存在于物理和计算宇宙中的东西，为人类服务。

嵌入 (Embeddings) 的概念

神经网络——至少目前的神经网络——基本上是基于数字的。因此，如果我们要使用它们来处理文本等内容，我们需要一种用数字来表示这些内容的方式。当然，我们可以像 ChatGPT 一样，为字典中的每个单词分配一个数字。但是还有一个重要的想法——例如，ChatGPT 的核心——超越了这个思路。那就是“嵌入”的概念。我们可以将嵌入视为一种尝试通过数字数组来表示某些东西的“本质”的方式——这些数字具有“附近事物”的性质，即“附近的事物”由相邻的数字表示。

例如，我们可以将单词嵌入视为一种尝试在某种“含义空间”中布置单词的方式，在这个空间中，具有某种“相似含义”的单词靠近嵌入中。实际使用的嵌入——例如在 ChatGPT 中——往往涉及大量数字的列表。但是，如果我们将它们投影到二维平面上，我们可以展示单词在嵌入中的分布：



是的，我们看到它在捕捉日常印象方面做得非常好。但是我们该如何构建这样的嵌入呢？粗略地说，想法是观察大量的文本（例如从网络中收集的 50 亿个单词），然后看看不同单词出现的“环境”有多相似。例如，“鳄鱼”和“鳄鱼”几乎可以互换地出现在其他相似的句子中，这意味着它们在嵌入中靠近。但是“萝卜”和“老鹰”通常不会出现在其他相似的句子中，因此它们在嵌入中会远离。

那么如何使用神经网络实现这种表示呢？我们来谈谈不是用于单词的嵌入，而是用于图像的嵌入。我们希望找到一种方式，以一些数字列表的方式来描述图像，使得我们认为“相似的图像”被赋予相似的数字列表。

我们如何判断是否应该“认为图像相似”？如果我们的图像是手写数字，我们可能会“认为两个图像相似”，如果它们是相同的数字。之前我们讨论过一个训练用于识别手写数字的神经网络。我们可以把这个神经网络看作是设置了一个最终输出，将图像放入 10 个不同的箱子，每个数字一个箱子。

但是，如果我们在最终“它是‘4’”的决策做出之前“拦截”正在发生的事情呢？我们可能会认为，在神经网络内部，有一些数字特征图像，表示“大多数像‘4’但有点像‘2’”之类的特征。这个想法是采集这样的数字，将其用作嵌入中的元素。

所以这就是概念。我们不是直接尝试表征“哪个图像靠近哪个图像”，而是考虑一个明确定义的任务（在这种情况下是数字识别），我们可以得到明确的训练数据 - 然后利用这个事实，在做这个任务时，神经网络隐含地做出了“近似度决策”。所以，我们从来没有必要明确讨论“图像的相似度”，我们只是在谈论一个具体的问题，即图像表示的数字是什么，然后“让神经网络”隐含地确定这意味着“图像的相似度”。

那么对于数字识别网络，这个概念如何更详细地工作呢？我们可以将网络看作由 11 个连续层组成，我们可以用这样的方式概括（激活函数显示为单独的层）：



最初，我们将实际图像通过第一层传入，由 2D 像素值数组表示。在最后一层中，我们从最后输出一个包含 10 个值的数组，我们可以认为它表示神经网络确定图像与数字 0 至 9 中的每个数字对应的程度。

将图像输入神经网络，输出层神经元的值如下：

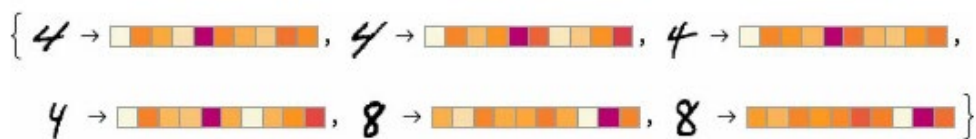
$$\{1.42071 \times 10^{-22}, 7.69857 \times 10^{-14}, 1.9653 \times 10^{-15}, 5.55229 \times 10^{-21}, 1., 8.33841 \times 10^{-14}, 6.89742 \times 10^{-17}, 6.52282 \times 10^{-19}, 6.51465 \times 10^{-12}, 1.97509 \times 10^{-14}\}$$

换句话说，神经网络现在对于这个图像是“非常确定”的它是一个 4——要得到输出“4”，我们只需要选出数值最大的神经元的位置。

但是，如果我们再向前看一步呢？网络的最后一个操作是 softmax，它试图“强制确定性”。但在它被应用之前，神经元的值是：

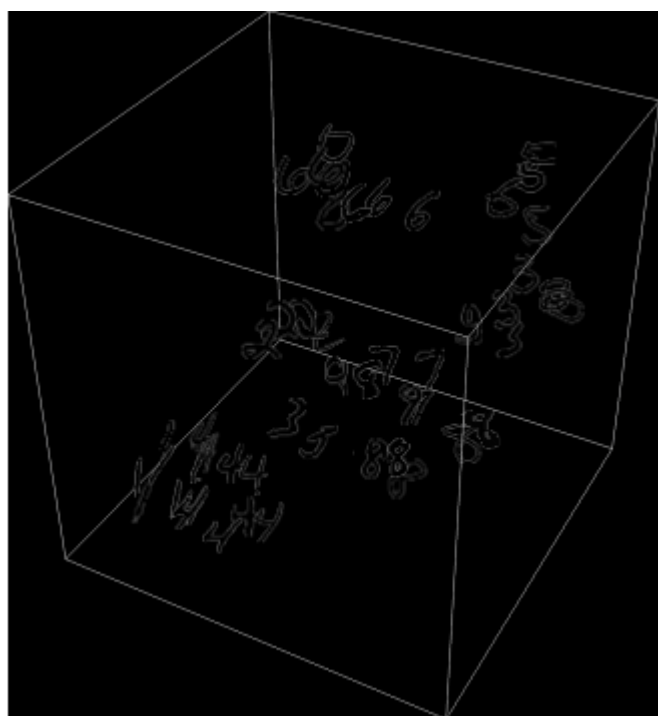
$$\{-26.134, -6.02347, -11.994, -22.4684, 24.1717, -5.94363, -13.0411, -17.7021, -1.58528, -7.38389\}$$

代表“4”的神经元仍然具有最高的数值。但其他神经元的值中也包含信息。我们可以预期，这个数字列表在某种程度上可以用于表征图像的“本质”——从而提供我们可以用作嵌入的内容。例如，在这里的每个4都具有略微不同的“签名”（或“特征嵌入”）——与8完全不同：



在这里，我们基本上使用 10 个数字来表征我们的图像。但通常最好使用更多数字。例如，在我们的手写数字识别网络中，我们可以通过接入前面一层来获得一个 500 个数字的数组。这可能是一个合理的数组，可以用作“图像嵌入”。

如果我们想要对手写数字的“图像空间”进行明确的可视化，我们需要“降维”，即通过将我们得到的 500 维向量投影到 3D 空间中实现：



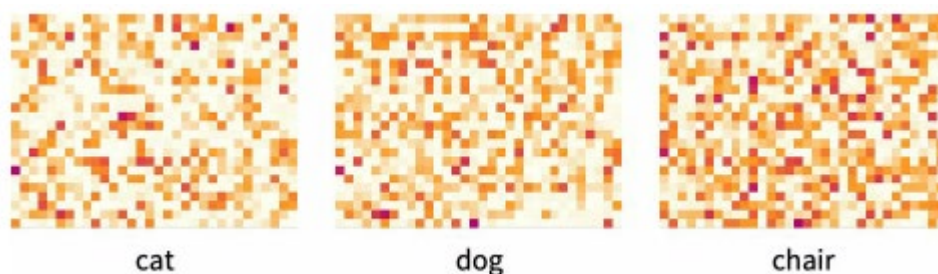
我们刚刚讨论了如何有效地创建图像的代表（和因此的嵌入），基本上是通过确定图像的相似性，即根据我们的训练集确定它们是否对应于相同的手写数字。如果我们有一个训练集，可以确定每个图像属于 5000 个常见类型的对象（例如猫、狗、椅子等），我们可以更普遍地做同样的事情。通过这种方式，我们可以创建一个以我们识别的常见对象为“锚点”的图像嵌入，但根据神经网络的行为进行“泛化”。重要的是，只要这种行为与我们人类感知和解释图像的方式相一致，这将最终成为一个“对我们来说似乎正确”的嵌入，并且在实践中用于执行“类人判断”的任务非常有用。

好的，那我们如何遵循同样的方法来找到单词的嵌入呢？关键是从一个有关单词的任务开始，我们可以很容易地进行训练。而标准的这种任务是“单词预测”。想象一下，我们被给出“the ___ cat”。根据一个大型的文本语料库（比如，Web 上的文本内容），可能会“填补空白”的不同单词有哪些概率？或者，给定“___ black ___”，不同“两侧单词”的概率是多少？

我们如何为神经网络设置这个问题？最终，我们必须用数字来表达所有东西。其中一种方法是为英语中的约 50,000 个常用单词分配一个唯一的数字。例如，“the”可能是 914，“cat”（前面有一个空格）可能是 3542。（这些是 GPT-2 使用的实际数字。）因此，对于“the ___ cat”问题，我们的输入可能是{914, 3542}。输出应该是一个包含大约 50,000 个数字的列表，实际上给出了每个可能的“填充”单词的概率。同样地，为了找到一个嵌入，我们希望在神经网络“达成结论”之前“拦截”其“内部”——然后捕捉发生在那里的数字列表，我们可以将其视为“表征每个单词”。

那么，这些表征看起来是什么样子的？在过去的 10 年里，已经开发了一系列不同的系统（如 word2vec、GloVe、BERT、GPT 等），每个系统都基于不同的神经网络方法。但最终，所有这些方法都采用单词，并通过数百到数千个数字的列表来表征它们。

这些“嵌入向量”的原始形式并不具有信息量。例如，这里是 GPT-2 为三个特定单词产生的原始嵌入向量的样子：



好的，那么我们如何按照同样的方法来找到单词的嵌入呢？关键是从一个关于单词的任务开始，我们可以很容易地进行训练。标准的任务是“单词预测”。想象一下，我们有“the ___ cat”。基于大量的文本语料库（比如网络上的文本内容），不同的单词“填空”的概率是多少？或者，给定“___ black ___”，不同的“侧面单词”的概率是多少？

我们如何为神经网络设置这个问题？最终我们必须用数字来表示一切。一种方法是为英语中约 50,000 个常见单词中的每个单词分配一个唯一的数字。因此，“the”可能是 914，“cat”（前面有一个空格）可能是 3542（这些是 GPT-2 实际使用的数字）。因此，对于“the ___ cat”问题，我们的输入可能是{914, 3542}。输出应该是一个包含约 50,000 个数字的列表，实际上为每个可能的“填充”单词提供概率。再次为了找到嵌入，我们想要“拦截”神经网络的“内部”，就在它“达到结论”之前，然后获取发生在那里的列表，我们可以认为这些数字“表征每个单词”。

那么这些特征长什么样子呢？在过去的 10 年中，已经开发出了一系列不同的系统（如 word2vec、GloVe、BERT、GPT 等），每个系统都基于不同的神经网络方法。但最终它们都将单词取样，通过数百到数千个数字的列表对其进行特征化。

在原始形式下，这些“嵌入向量”相当无信息。例如，这里是 GPT-2 为三个特定单词生成的原始嵌入向量：如果我们计算这些向量之间的距离，那么我们就可以找到单词的“相似度”。稍后我们将更详细地讨论这样的嵌入的“认知”意义。但是现在的主要观点是，我们有一种有用的方法将单词转换为“神经网络友好”的数字集合。

但实际上，我们可以更进一步，不仅可以通过数字集合来表征单词，还可以为单词序列，甚至整个文本块做到这一点。在 ChatGPT 内部，它就是这样处理的。它接受到目前为止的文本，并生成一个嵌入向量来代表它。然后它的目标是找到可能出现的不同单词的概率。并且，

ChatGPT 内部

好了，我们终于准备好讨论 ChatGPT 内部是什么了。毫无疑问，它最终是一个巨大的神经网络——目前是一个拥有 1750 亿个权重的 GPT-3 网络版本。在许多方面，这是一个与我们讨论过的其他神经网络非常相似的神经网络。但是，这是一个专门用于处理语言的神经网络，其最显著的特点是神经网络架构中的“变形器”。

在我们上面讨论的第一个神经网络中，任何给定层的每个神经元基本上都与前一层的每个神经元连接（至少有一些权重）。但是，如果处理具有特定已知结构的数据，则这种完全连接的网络可能过于臃肿。因此，例如，在处理图像的早期阶段，通常使用所谓的卷积神经网络（“convnets”），其中神经元实际上是布置在类似于图像中的像素的网格上，并且仅与网格上附近的神经元相连。

变形器的想法是对组成文本片段的令牌序列进行类似的操作。但是，变形器不是仅定义一段可以进行连接的序列，而是引入了“注意力”概念和“更多地关注”序列的某些部分的想法。也许有一天，通过训练就可以启动通用的神经网络并完成所有自定义工作，但至少目前来说，实践中“模块化”事物似乎是至关重要的——正如变形器所做的那样，也可能是我们的大脑所做的。

好的，那么 ChatGPT（或者说，它所基于的 GPT-3 网络）实际上是做什么的呢？请记住，它的总体目标是基于它从训练数据中（包括查看来自 Web 等的数十亿个文本页面）所看到的，以“合理的方式”继续文本。因此，在任何给定时刻，它都有一定量的文本，并且其目标是选择适当的下一个令牌来添加。

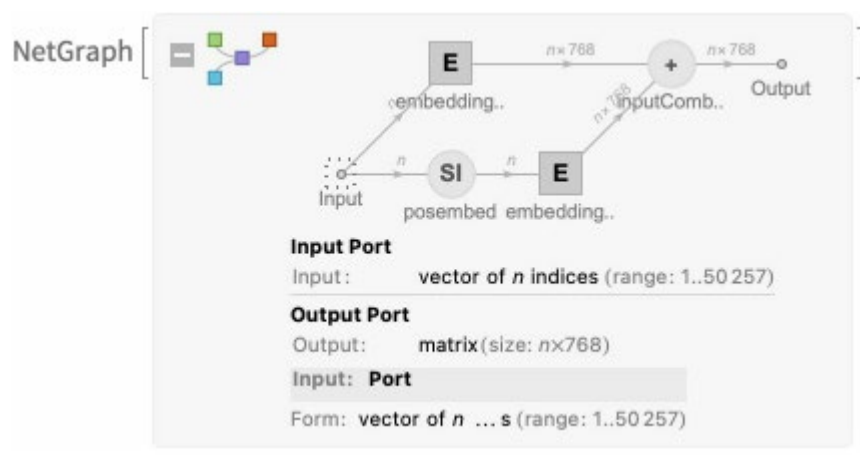
它有三个基本阶段。首先，它获取到迄今为止文本所对应的令牌序列，并找到表示这些令牌的嵌入（即一组数字）。然后，它在此嵌入上操作，以“标准神经网络方式”进行操作，值会“涟漪”到网络的连续层中，以生成一个新的嵌入（即一个新的数字数组）。然后，它获取此数组的最后一部分，并生成一个大约有 50,000 个值的数组，这些值会变成

不同可能的下一个令牌的概率。（是的，碰巧用到的令牌数与英语中常用单词数大致相同，尽管其中仅有约 3000 个令牌是完整单词，其余都是片段。）

关键的一点是，管道中的每个部分都由神经网络实现，其权重由网络的端到端训练确定。换句话说，实际上除了整体架构之外，没有任何东西是“显式设计”的；一切都只是从训练数据中“学习”的。

然而，在建立架构的方式上有大量的细节，反映了各种经验和神经网络学说。虽然这绝对有点深入，但我认为谈论其中一些细节是有用的，至少可以了解到构建类似 ChatGPT 的东西需要多少工作。

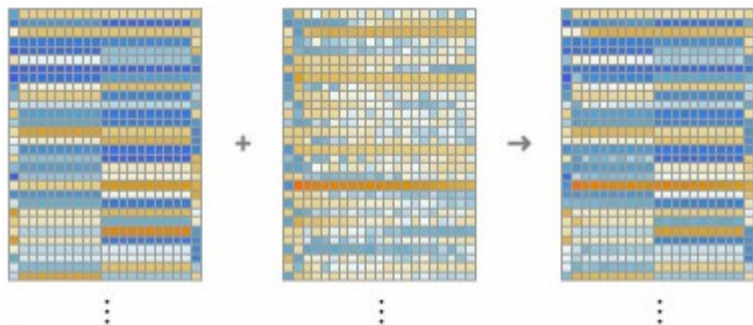
首先是嵌入模块。下面是 GPT-2 的它的一个示意性 Wolfram 语言表示：



该模块的输入是 n 个令牌（如前一部分所示，由 1 到大约 50,000 的整数表示）。其中每个令牌都通过一个单层神经网络转换为一个嵌入向量（对于 GPT-2 长度为 768，对于 ChatGPT 的 GPT-3 长度为 12,288）。同时，还有一个“次要路径”，它获取令牌的整数位置序列，并从这些整数创建另一个嵌入向量。最后，将来自令牌值和令牌位置的嵌入向量相加，以生成来自嵌入模块的最终嵌入向量序列。

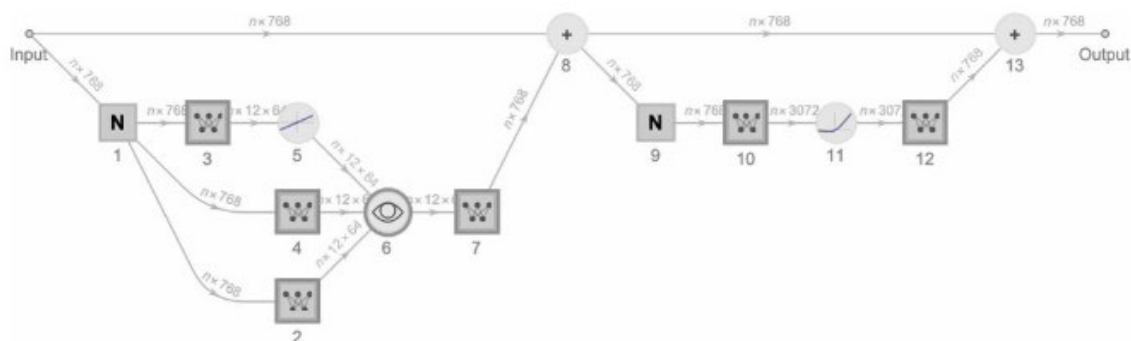
为什么只需将令牌值和令牌位置嵌入向量相加呢？我认为这并没有特别的科学性。只是尝试了许多不同的方法，这种方法似乎行之有效。而且神经网络的传统认为是，只要设置大致正确，通常可以通过进行足够的训练来解决细节问题，而无需真正“理解神经网络已经配置自己的工程级别的情况”。

以下是嵌入模块的示例，它对字符串 hello hello hello hello hello hello hello hello hello hello bye bye bye bye bye bye bye bye bye 进行操作：



向量中的元素以每个令牌的嵌入向量显示在页面上，并且在页面上，我们首先看到一组“hello”嵌入向量，然后是一组“bye”嵌入向量。上面的第二个数组是位置嵌入，其略微随机的结构仅仅是“被学习到的”（在本例中是在 GPT-2 中学习到的）。

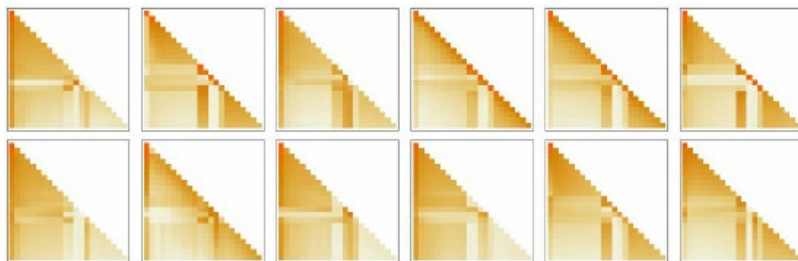
好的，嵌入模块之后是变压器的“主要事件”：一系列所谓的“注意力块”（GPT-2 为 12 个，ChatGPT 的 GPT-3 为 96 个）。这一切非常复杂，让人想起典型难以理解的大型工程系统，或者说生物系统。但无论如何，这里是单个“注意力块”的示意图（适用于 GPT-2）：



在每个这样的注意力块中，都有一组“注意力头”（GPT-2 为 12 个，ChatGPT 的 GPT-3 为 96 个），每个头都独立地在嵌入向量的不同值块上运行。（是的，我们不知道将嵌入向量分割成不同的部分，或者不同的部分代表什么含义是个好主意的原因，这只是那些“被发现有效”的事情之一。）

好的，那么注意力头是做什么的呢？基本上，它们是一种“回顾”标记序列（即迄今为止生成的文本），并以对寻找下一个标记有用的形式“打包过去”。在上面的第一节中，我们谈到使用 2-gram 概率根据其直接前任选择单词。转换器中的“注意力”机制允许“关注”甚至更早的单词，从而潜在地捕捉诸如动词如何引用在句子中出现许多单词之前的名词的方式。

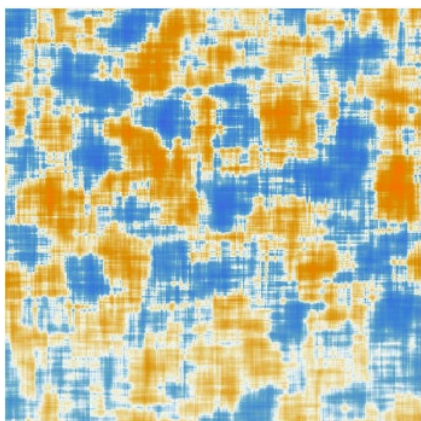
在更详细的层面上，注意力机制的作用是重新组合与不同标记相关的嵌入向量块，具有一定的权重。例如，GPT-2 中第一个注意力块中的 12 个注意力头对于上面的“hello, bye”字符串具有以下“重新组合权重”的模式（“回望整个标记序列的模式”）：



经过注意力头的处理，结果的“重新加权嵌入向量”（在 GPT-2 中为长度 768，在 ChatGPT 的 GPT-3 中为长度 12,288）被传递到一个标准的“全连接”神经网络层。很难掌握这一层在做什么。但是，这里是它使用的 768×768 权重矩阵的绘图（这里是 GPT-2）：



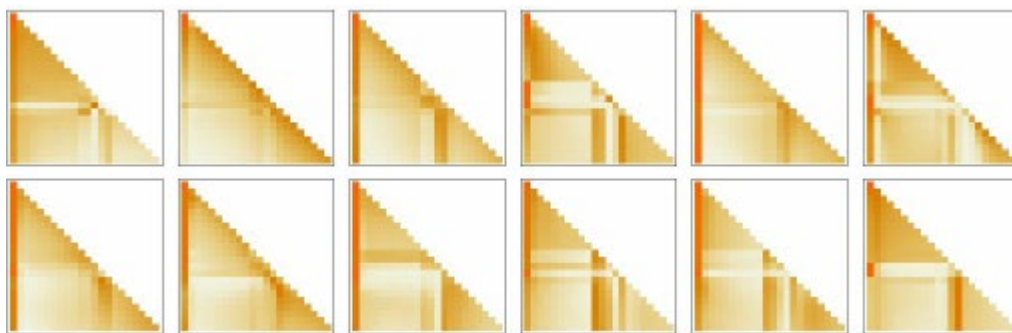
取 64×64 移动平均值，会出现一些（随机漫步般的）结构：



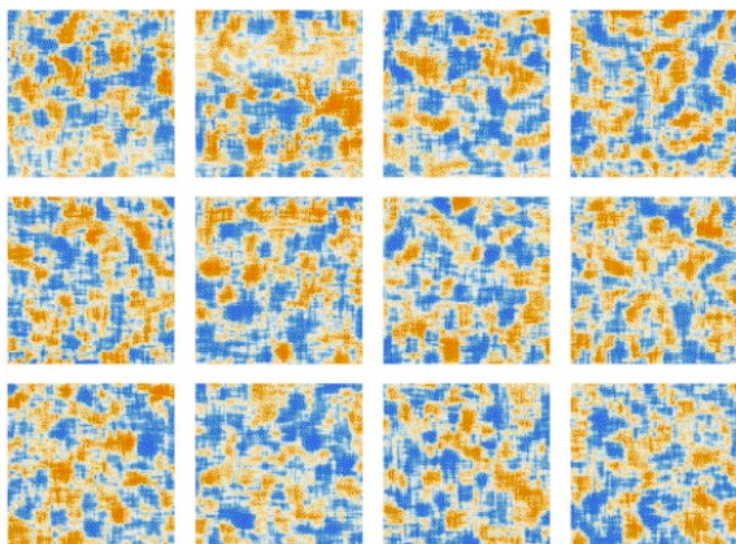
什么决定了这个结构？最终，这可能是人类语言特征的一些“神经网络编码”。但就目前而言，这些特征可能是未知的。实际上，我们正在“打开 ChatGPT 的大脑”（或至少是 GPT-2），并发现，是的，那里很复杂，我们不理解它——尽管最终它产生了可识别的人类语言。

经过一个注意力块的处理后，我们得到了一个新的嵌入向量，然后它会被连续地传递到其他注意力块中（GPT-2 有 12 个注意力块，GPT-3 有 96 个）。每个注意力块都有自己

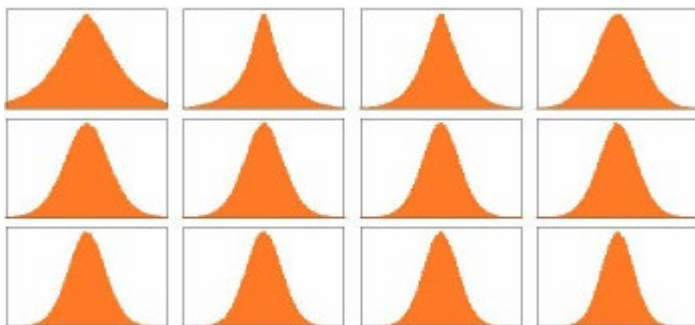
特定的“注意力”和“全连接”权重模式。下面是对于第一个注意力头，“hello, bye”输入的注意力权重序列：



以下是（移动平均后的）全连接层的“矩阵”：



有趣的是，尽管这些不同注意力块中的“权重矩阵”看起来相似，但是权重大小的分布可能会有所不同（而且不总是高斯分布）：



那么经过所有这些注意力块后，变压器的净效应是什么？它的本质就是将序列中的原始嵌入向量转换为最终集合。而 ChatGPT 的特定工作方式是挑选这个集合中的最后一个嵌入向量，并将其“解码”为下一个标记的概率列表。

这就是 ChatGPT 内部的概述。尽管存在许多不可避免的略有任意性的“工程选择”，但它似乎很复杂，但最终涉及的元素非常简单。因为最终我们处理的只是一个由“人造神经元”组成的神经网络，每个神经元都执行着一个简单的操作，即接受一组数值输入，然后将它们与特定的权重相结合。

ChatGPT 的原始输入是一个数字数组（至今为止标记的嵌入向量），当 ChatGPT“运行”以生成新标记时，这些数字会“涟漪”通过神经网络的层，每个神经元都会“做它的事情”并将结果传递到下一层的神经元。没有循环或“回退”。一切都只是通过网络“前馈”。

这与典型计算系统（如图灵机）非常不同，在这种系统中，结果会由相同的计算元素重复“重新处理”。在这里，至少在生成给定的输出标记时，每个计算元素（即神经元）仅使用一次。

但在某种程度上，即使在 ChatGPT 中，仍然存在一种“外部循环”重复使用计算元素。因为当 ChatGPT 要生成新标记时，它总是“读取”（即将其作为输入）前面的所有标记，包括 ChatGPT 自己之前“写入”的标记。我们可以将这种设置视为意味着 ChatGPT 在其最外层至少涉及一个“反馈循环”，尽管每次迭代都作为出现在它生成的文本中的标记显式可见。

但是让我们回到 ChatGPT 的核心：重复使用以生成每个标记的神经网络。在某种程度上，它非常简单：整个相同的人工神经元集合。网络的某些部分只包含（“全连接”）神经元层，其中给定层上的每个神经元都与之前的层上的每个神经元连接（具有一些权重）。但特别是在其变形器架构中，ChatGPT 具有具有更多结构的部分，其中仅连接不同层上的特定神经元。（当然，人们仍然可以说“所有神经元都连接在一起”，但有些神经元权重为零。）

此外，ChatGPT 中的神经网络还存在一些不能被视为仅由“同构”层组成的方面。例如，正如上面的标志性总结所示，在注意块中有“多个拷贝”的地方，每个都通过不同的“处理路径”进行处理，潜在地涉及不同数量的层，然后才重新组合。但虽然这可能是正在发生的事情的一种方便的表示，但在原则上，总是可以想象“密集填充”层，但只是使一些权重为零。

如果查看通过 ChatGPT 的最长路径，则涉及约 400 个（核心）层-在某些方面不算很多。但有数百万个神经元-总共有 1750 亿个连接，因此有 1750 亿个权重。需要意识到的一件事是，每次 ChatGPT 生成一个新标记时，它都必须进行涉及每个权重的计算。在实现上，这些计算可以通过高度并行的数组操作“按层”组织起来，方便地在 GPU 上完成。但对于每个生成的标记，仍然必须完成 1750 亿个计算（最后还要多一些）-因此，不出所料，使用 ChatGPT 生成一段长文本可能需要一些时间。

但最终，值得注意的是，所有这些操作-尽管它们各自很简单-却可以共同完成这样一个出色的“类人”文本生成工作。必须再次强调（至少就我们目前所知），没有“终极理论原因”说明为什么会有像这样的事情发生。实际上，正如我们将要讨论的那样，我认为我们必须将其视为一项-可能令人惊讶的-科学发现：在像 ChatGPT 的神经网络中，以某种方式捕捉到了人类大脑在生成语言方面所能做的本质。

ChatGPT 的训练

好的，现在我们已经概述了 ChatGPT 设置后的运作方式。但是它是如何设置的呢？它的 1750 亿个权重是如何确定的呢？基本上，它们是通过基于人类编写的大量文本语料库进行大规模训练的结果，这些文本来自网络、书籍等。正如我们所说，即使有了所有这些训练数据，神经网络能够成功地生成“人类风格”的文本也绝非显而易见。并且，再一次强调，似乎需要一些详细的工程要素才能实现这一点。但 ChatGPT 的惊人之处——和发现——在于这是可能的。实际上，一个“只有”1750 亿个权重的神经网络可以构建出一个“合理的模型”，模拟人类写作的文本。

现代社会有许多以数字形式存在的人类写作文本。公共网络上至少有数十亿个人类编写的网页，总共可能有万亿字的文本。如果包括非公共网页，这个数字可能要大 100 倍。到目前为止，已经提供了超过 500 万本数字化的书籍（共有大约 1 亿本出版过的书籍），提供了大约 1000 亿个单词的文本。这还没有提到视频中的语音衍生文本等。作为个人的比较，我一生中发表的总文字量不到 300 万字，过去 30 年我写了大约 1500 万字的电子邮件，总共打了大约 5000 万字的字——仅在过去的几年里，我就在直播中说了超过 1000 万个字。是的，我将使用所有这些来训练一个机器人。

好的，那么，有了这些数据，怎么训练一个神经网络呢？基本过程就像我们之前讨论的简单例子一样。你展示一批例子，然后调整网络中的权重，以最小化网络在这些例子上产生的误差（“损失”）。“反向传播”误差最主要的开销在于每次这样做时，网络中的每个权重通常都会至少微调一点，而需要处理的权重数量就是很多。（实际的“反向计算”通常只比前向计算更难一点点。）

通过现代 GPU 硬件，可以轻松地并行计算成千上万个例子的结果。但是，在实际更新神经网络的权重时，目前的方法基本上需要一批一批地完成。（是的，这可能是实际大脑在计算和内存元素上至少有一个架构优势的地方。）

即使在我们之前讨论的学习数值函数的看似简单的情况下，我们发现我们通常需要使用数百万个例子才能成功地从头开始训练一个网络。那么，要训练一个“类人的语言”模型，我们需要多少例子呢？在实践中，ChatGPT 成功地在几百亿个单词的文本上进行了训练。

ChatGPT 曾多次接收了同一些文本，有些则仅接收了一次。但不知道在这么大的语料库中学习语言的神经网络需要多大规模的网络结构才能“学得好”？同样地，我们目前没有一个基本的理论方法可以回答这个问题。最终，正如我们将在下面进一步讨论的那样，人类语言和人们通常用它说的话可能存在一定的“总算法内容”。但接下来的问题是神经网络在实现基于该算法内容的模型时的效率有多高。同样，我们不知道，尽管 ChatGPT 的成功表明它的效率是合理的。

最终，我们可以指出的是，ChatGPT 使用了几千亿个权重来完成它的任务，这个数量与它所接受的训练数据的总词数（或令牌数）相当。在某些方面，令人惊讶的是（尽管

在 ChatGPT 的较小模拟中也有经验观察到这一点），似乎能够很好地工作的“网络大小”与“训练数据大小”如此相近。毕竟，ChatGPT 内部显然不是“直接存储”来自网络、书籍等所有文本的内容。因为实际上 ChatGPT 内部是由一堆数字组成的，这些数字有不到 10 位数字的精度，它们是所有这些文本的聚合结构的某种分布式编码。

换个方式来说，我们可以问人类语言和通常使用它说的话的“有效信息内容”是多少。有语言的原始语料库，还有 ChatGPT 神经网络中的表示。这种表示很可能远非“算法上最小”的表示方式（如下面所述）。但这是一种神经网络可以方便地使用的表示方式。在这种表示中，训练数据似乎基本上很少经过“压缩”；每个训练数据词的信息内容基本上只需要略少于一个神经网络权重来传输。

当我们运行 ChatGPT 生成文本时，我们基本上需要使用每个权重一次。因此，如果有 n 个权重，则我们需要大约 n 个计算步骤，虽然实际上其中许多步骤通常可以在 GPU 中并行处理。但是，如果我们需要大约 n 个词的训练数据来设置这些权重，那么根据我们上面所说的，我们可以得出结论，我们需要约 n^2 个计算步骤来训练网络，这就是为什么当前方法需要耗费数十亿美元来进行培训。

除了基本的训练之外

训练 ChatGPT 的大部分工作都花在“展示大量现有文本”方面。但是结果表明还有另一个似乎非常重要的方面。

一旦 ChatGPT 从它展示的原始文本语料库中完成“原始训练”，内部神经网络就准备好开始生成自己的文本，从提示中继续等等。虽然这样生成的结果通常看起来合理，但尤其是对于较长的文本，它们往往会“偏离”常常是非常不像人类的方式。这不是通过对文本进行传统统计方法就能很容易地检测出来的。但是，实际阅读文本的人很容易注意到这一点。

ChatGPT 构建的关键思想之一是，在“被动阅读”像网络等事物之后，进行另一步操作：实际的人类互动 ChatGPT，查看它产生的内容，并在实际上对“如何成为一个好的聊天机器人”给予反馈。但是，神经网络如何利用这个反馈？第一步只是让人类评估神经网络的结果。然后建立另一个神经网络模型，试图预测这些评级。但是现在可以在原始网络上运行这个预测模型，类似于损失函数，实际上允许网络通过人类反馈来进行“调整”。实践中的结果似乎对于该系统成功产生“类人”的输出有很大的影响。

总的来说，有趣的是，为了让“最初训练”的网络在特定方向上有用地行动，需要的“刺激”非常少。人们可能认为为了让网络的行为就像“学习了一些新的东西”，必须运行训练算法，调整权重等等。

然而，情况并非如此。相反，似乎只需在提示中告诉 ChatGPT 一次某些信息，它就能成功地在生成文本时利用它。再次强调，这个方法的成功是理解 ChatGPT“真正所做的事情”以及它如何与人类语言和思维结构相关的重要线索。

在这种情况下，确实有一些类似人类的东西：至少在经过了所有的预训练之后，你只需告诉它一次就可以“记住”它——至少“足够长时间”以便使用它生成一段文本。那么这样做时发生了什么呢？可能是“你可能告诉它的一切信息已经存在于它的某个地方”——你只是把它引导到正确的位置。但这似乎不可信。相反，更有可能的是，是的，这些元素已经在那里了，但具体的细节由像“这些元素之间的轨迹”之类的东西定义，而你所做的就是引入这些轨迹。

实际上，与人类类似，如果你告诉它一些奇怪和意外的事情，完全不符合它已知的框架，它似乎无法成功地“整合”这些信息。只有当它基本上在已有的框架上以相当简单的方式行驶时，才能够“整合”它。

需要再次指出的是，神经网络无法避免“算法限制”。如果给它“表层”形式的规则，例如“这个变成那个”，神经网络很可能能够良好地表示和重现这些规则，并且它已经从语言中“学习到的”东西将给它一个即时的模式可供跟随。但是，如果尝试给它一个涉及多个潜在的不可约步骤的“深度”计算规则，它将无法正常工作。（请记住，在每一步中，它总是仅在其网络中“向前传递数据”，除了生成新令牌外不会循环。）

当然，网络可以学习特定的“不可约”计算答案。但是一旦存在组合数量的可能性，就不会有这种“表格查找式”的方法可行。因此，就像人类一样，神经网络需要“伸出手”并使用实际的计算工具。（是的，Wolfram|Alpha 和 Wolfram 语言非常适合，因为它们已经被构建为“谈论世界中的事物”，就像语言模型神经网络一样。）

什么让 ChatGPT 能够工作？

人类语言和生成它所涉及的思维过程，一直被认为是复杂性的巅峰。事实上，人类的大脑只有大约 1000 亿个神经元（可能有 1 万亿个连接），却能够创造出如此复杂的语言。也许有人会想，大脑除了神经元网络还有其他未被发现的物理层面。但现在有了 ChatGPT，我们有了一个新的信息：我们知道一个纯粹的、具有与大脑神经元数量相当的人工神经网络，能够出奇地好地生成人类语言。

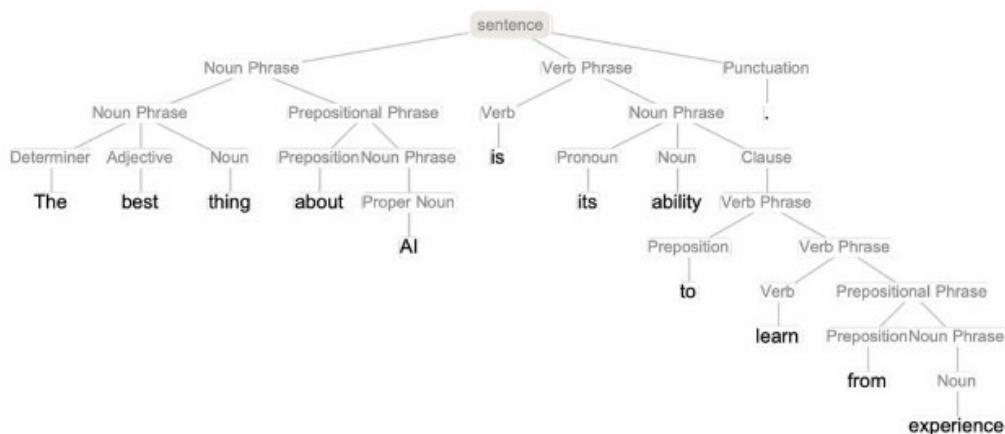
是的，这仍然是一个庞大而复杂的系统，大约有与世界上当前可用的文字数量相当的神经网络权重。但在某种程度上，仍然很难相信语言的所有丰富性和它所能谈论的事物可以被封装在如此有限的系统中。部分原因是，计算过程可以在其基本规则简单的情况下，通过放大系统的表现复杂性来大幅度提高系统的复杂性（这在规则 30 的例子中首次表现出来）。但正如我们上面讨论的那样，ChatGPT 中使用的神经网络类型通常是特别构建的，以限制这种现象和与之相关的计算不可化简性，以便使它们的训练更易于理解。

那么，像 ChatGPT 这样的东西是如何实现对语言的理解的呢？基本的回答，我认为，是语言在根本层面上某种程度上比看起来要简单。这意味着，即使是具有最终简单的神经网络结构的 ChatGPT 也能够成功地“捕捉到”人类语言及其背后的思维的本质。此外，在训练过程中，ChatGPT 以某种方式“隐含地发现”了使这一切成为可能的语言（和思考）的规律性。

我认为，ChatGPT 的成功为我们提供了一项基本而重要的科学证据：它表明我们可以期望发现重要的新“语言法则”，以及实际上是“思考法则”。在像神经网络这样的 ChatGPT 中，这些法则充其量是隐含的。但是，如果我们能够以某种方式使这些法则变得明确，那么就有可能以更直接、更高效和更透明的方式执行 ChatGPT 所做的各种事情。

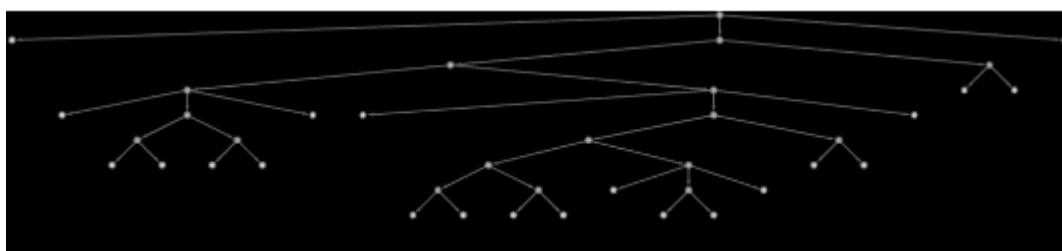
但是，那么这些规则可能是什么样子呢？最终，它们必须给我们提供一些关于如何组合语言以及我们用它说的事情的指导。稍后我们将讨论“查看 ChatGPT 内部”可能能够为我们提供一些提示，以及从构建计算语言方面得出的认识提供了一条前进的道路。但首先，让我们讨论两个长期已知的“语言法则”的例子，以及它们与 ChatGPT 的运作方式的关系。

第一个是语言的句法结构。语言不仅仅是随机的词语组合。相反，不同种类的词语可以如何组合有（相当）明确的语法规则：例如，在英语中，名词可以由形容词修饰，并跟随动词，但通常两个名词不能紧挨着。这种语法结构可以（至少近似地）通过定义“解析树”如何组合的一组规则来捕捉到：



ChatGPT 没有任何明确的关于这些规则的“知识”。但是在它的训练过程中，它隐含地“发现”了它们，然后似乎很擅长遵循它们。那么这是如何实现的呢？在“大局”层面上不是很清楚。但是为了获得一些见解，看一个更简单的例子可能是有益的。

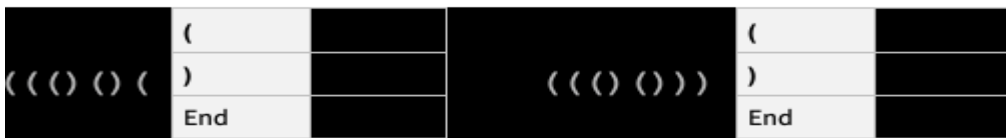
考虑一个由（和）序列组成的“语言”，语法规定括号必须平衡，如下面的解析树所示：



我们可以训练一个神经网络生成“语法正确”的括号序列吗？在神经网络中处理序列有不同的方式，但让我们使用转换器网络，就像 ChatGPT 一样。并且，给定一个简单的转换器网络，我们可以开始将语法正确的括号序列作为训练示例输入。一个细节（实际上也出现在 ChatGPT 生成人类语言的过程中）是，除了我们的“内容标记”（这里是“(”和“)”）外，我们还必须包括一个“End”标记，用于指示输出不应继续（即对于 ChatGPT，到达了“故事的结尾”）。

如果我们使用只有一个具有 8 个头和长度为 128 的特征向量的注意力块的转换器网络，则似乎不可能使它学习括号语言的大部分知识。但是使用 2 个注意力块，学习过程似乎会收敛，至少在给出了大约 1000 万个示例后会收敛（正如转换器网络通常所遇到的那样，给出更多的示例只会降低它的性能）。

因此，通过这个网络，我们可以执行 ChatGPT 所做的类比，并询问下一个标记应该是什么的概率，例如在括号序列中：



在第一种情况下，网络“非常确定”序列不能在此结束，这是好的，因为如果它确实结束了，括号将不平衡。然而，在第二种情况下，它“正确地识别出”序列可以在此结束，尽管它也“指出”可能会“重新开始”，放下一个“(”，然后跟随一个“)”。但是，糟糕的是，即使通过费力训练了大约 400,000 个权重，它也说下一个标记“)”的概率是 15%——这是不对的，因为这必然会导致不平衡的括号。

如果我们要求网络提供越来越长的“(”序列的最高概率完成情况，我们将得到以下结果：

```
( )
(( ))
((( )))
(((( ))))
(((( ( ))) )
(((( (( )))))
(((( ((( ))))) )
(((( ((( ( ))) ))) )
(((( ((( ( ( ))) ))) )
(((( ((( ( ( ( ))) ))) )
(((( ((( ( ( ( ( ))) ))) )
(((( ((( ( ( ( ( ( ))) ))) )
(((( ((( ( ( ( ( ( ( ))) ))) ) (unbalanced)
(((( ((( ( ( ( ( ( ( ( ))) ))) ) (unbalanced)
(((( ((( ( ( ( ( ( ( ( ( ))) ))) ) )
(((( ((( ( ( ( ( ( ( ( ( ( ))) ))) ) )
(((( ((( ( ( ( ( ( ( ( ( ( ( ))) ))) )
(((( ((( ( ( ( ( ( ( ( ( ( ( ( ))) ))) ) (unbalanced)
(((( ((( ( ( ( ( ( ( ( ( ( ( ( ( ))) ))) ) (unbalanced)
(((( ((( ( ( ( ( ( ( ( ( ( ( ( ( ))) ))) )
(((( ((( ( ( ( ( ( ( ( ( ( ( ( ( ( ))) ))) ) )
```

是的，在一定长度范围内，网络做得很好。但是，它开始失败了。这是神经网络（或机器学习总体上）在“精确”情况下通常会出现的一种典型情况。人类“可以一眼解决”的情况，神经网络也可以解决。但是对于需要进行“更算法化”的情况（例如显式计算括号是否关闭），神经网络往往因某种原因“计算过于浅显”，难以可靠地执行。（顺便说一句，即使是完整的 ChatGPT 目前也难以在长序列中正确匹配括号。）

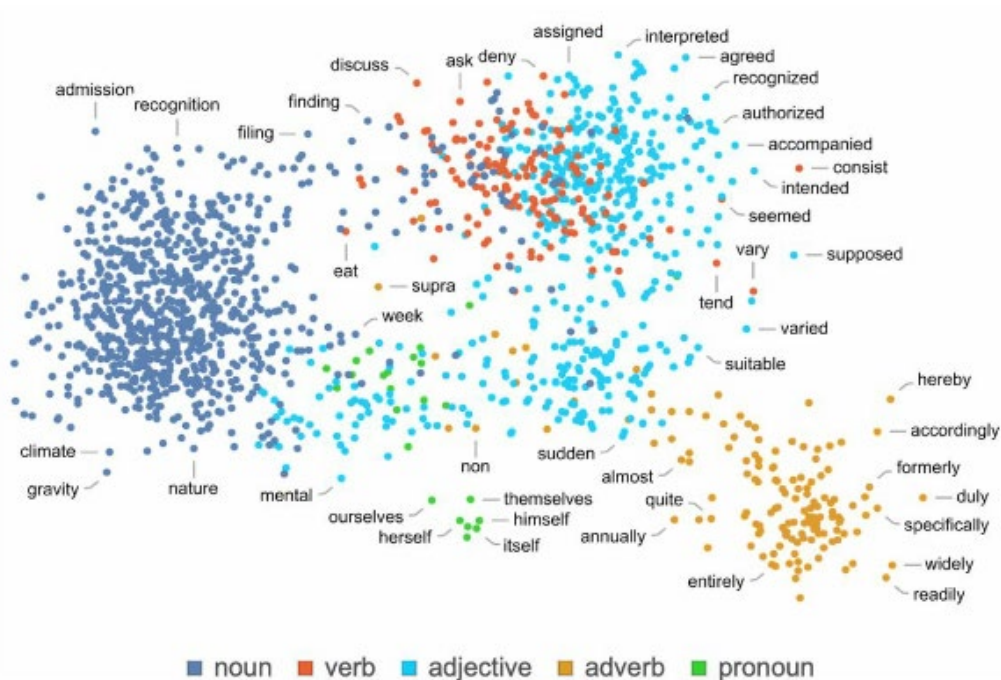
那么这对 ChatGPT 和英语这样的语言的语法意味着什么呢？括号语言是“朴素”的，更多的是一个“算法式的故事”。但在英语中，基于单词的局部选择和其他提示，能够“猜测”什么在语法上能够适应是更加现实的。是的，神经网络在这方面表现得更好，即使它可能会错过一些“形式上正确”的情况，这些情况人类也可能会忽略。但主要观点是，语言存在总体的语法结构，其中有所有规则，这在某种程度上限制了神经网络必须学习的“多少”。关键的“自然科学式”的观察是，像 ChatGPT 中那样的神经网络的转换器架构似乎能够成功地学习嵌套树状的语法结构，这种结构似乎以某种逼近方式存在于所有人类语言中。

语法提供了一种对语言的限制。但显然还有更多的限制。像“Inquisitive electrons eat blue theories for fish”这样的句子在语法上是正确的，但这不是人们通常希望说的话，如果 ChatGPT 生成这样的句子，也不会被认为是成功的，因为，嗯，就其词义而言，它基本上是毫无意义的。

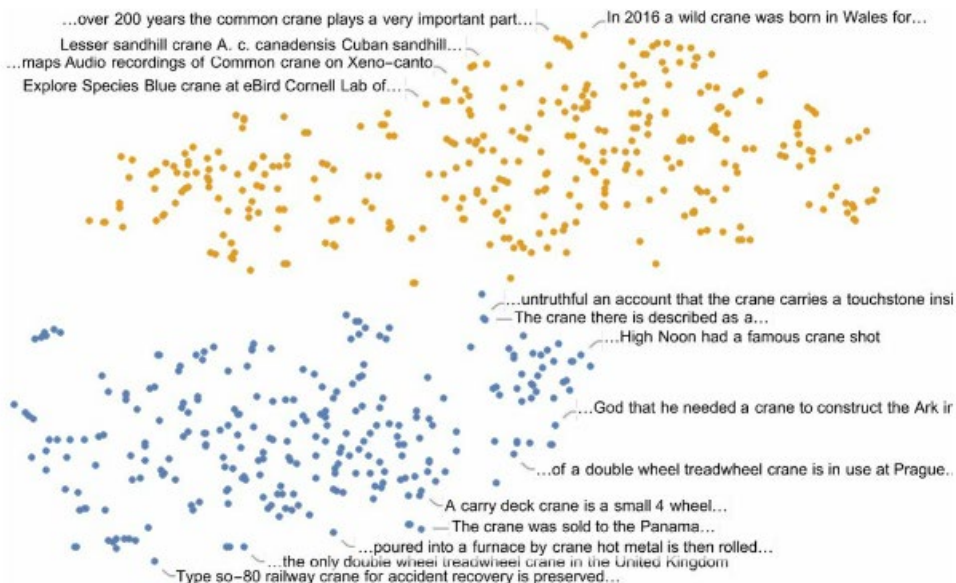
但是否有一种通用的方法来判断一句话是否有意义？传统上并没有总体理论。但可以认为 ChatGPT 在训练过程中隐含地“开发了一个理论”，在此过程中受到来自网络等数十亿（可能是有意义的）句子的影响。

这个理论可能是什么样子呢？好吧，有一小部分的理论已经存在了两千年，那就是逻辑。在亚里士多德发现的三段论形式中，逻辑基本上是一种说法，即遵循某些模式的句子是合理的，而其他的则不合理。因此，例如，说“所有的 X 都是 Y。这不是 Y，所以它不是 X”（如“所有的鱼都是蓝色的。这不是蓝色的，所以它不是鱼。”）是合理的。就像人们可以有些异想天开地想象亚里士多德通过大量修辞的例子“机器学习式”地发现了三段论逻辑一样，我们也可以想象，在 ChatGPT 的训练中，它通过查看网上的大量文本等方式能够“发现三段论逻辑”。（是的，尽管因此可以期望 ChatGPT 生成包含基于三段论逻辑等的“正确推理”的文本，但当涉及到更复杂的形式逻辑时，它将因为与匹配括号时相同的原因而失败。）

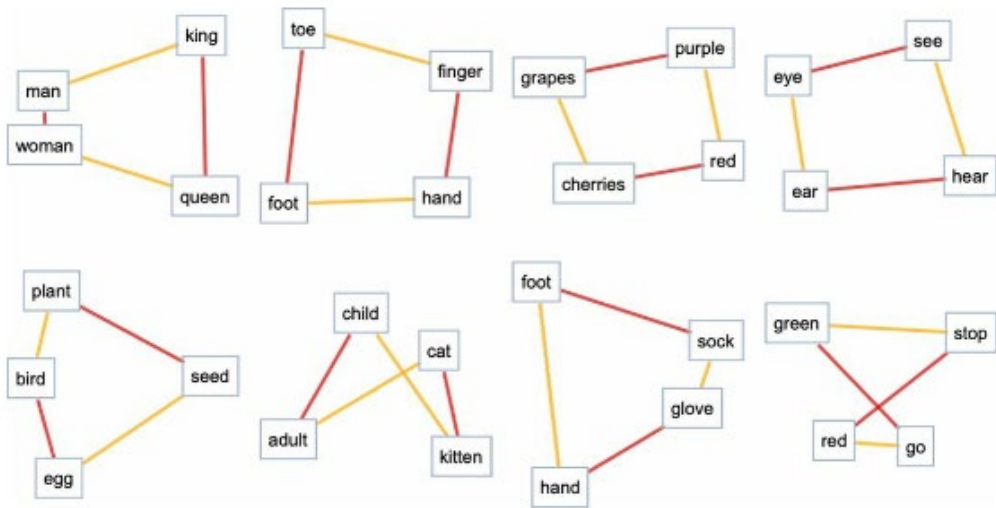
但除了逻辑这个狭隘的例子，关于如何系统地构建（或识别）即使是可能有意义的文本，还能说些什么呢？是的，像“Mad Libs®”这样的东西使用非常特定的“词组模板”。但不知何故，ChatGPT 隐含地有一种更通用的方法来做到这一点。也许除了“当你拥有 1750 亿个神经网络权重时，某种情况下会发生”以外，我们无法进一步解释这是如何做到的。但我强烈怀疑有一个更简单、更强大的故事。



当然，一个单词通常不只有“一个意思”（或者不一定对应于一个词性）。通过观察包含单词的句子在特征空间中的分布，我们通常可以“区分出”不同的含义，比如这里的单词“鹤”（是指鸟还是机器？）。

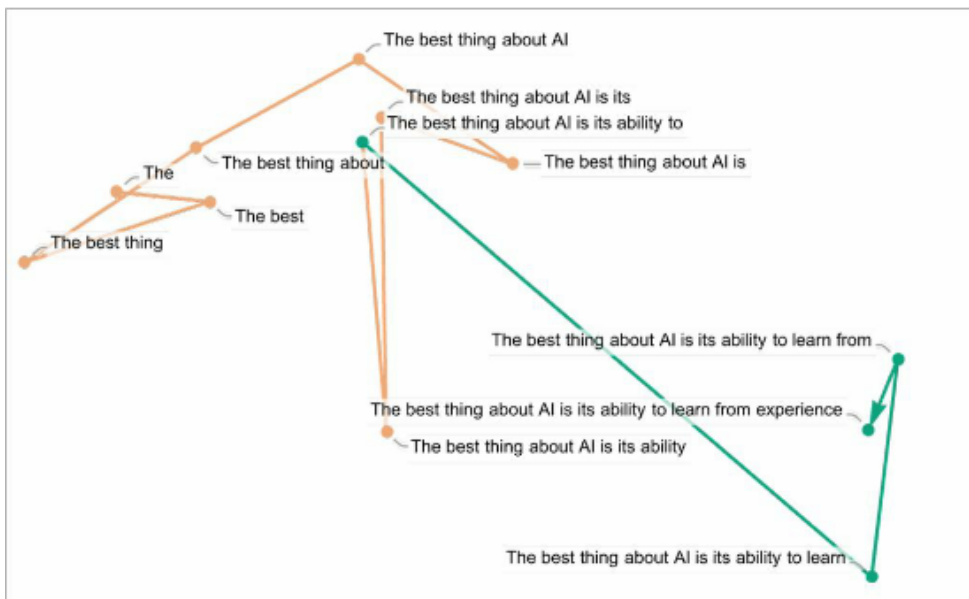


因此，我们可以认为这个特征空间将“语义相似的单词”放在空间中彼此接近的位置。但我们能够在这个空间中识别什么样的附加结构？例如，是否存在某种“平行传输”的概念，以反映空间的“平坦性”？一种了解这个问题的方法是看一下类比：



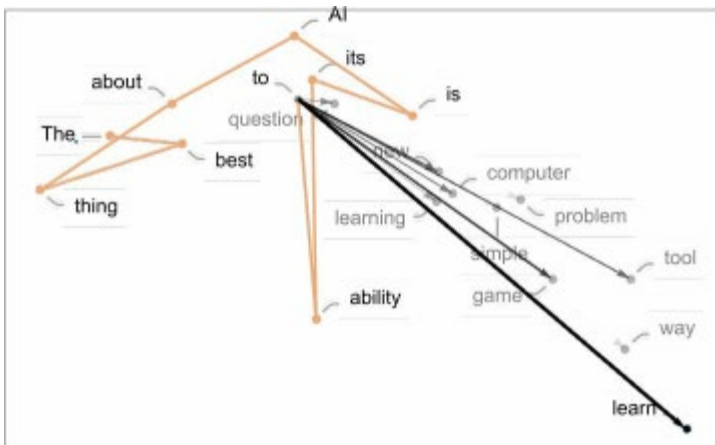
即使我们将其投影到二维平面上，通常也可以看到“平坦性的提示”，尽管这并非普遍存在。

那么，轨迹呢？我们可以查看 ChatGPT 的提示在特征空间中的轨迹，然后看一下 ChatGPT 如何继续这一轨迹：

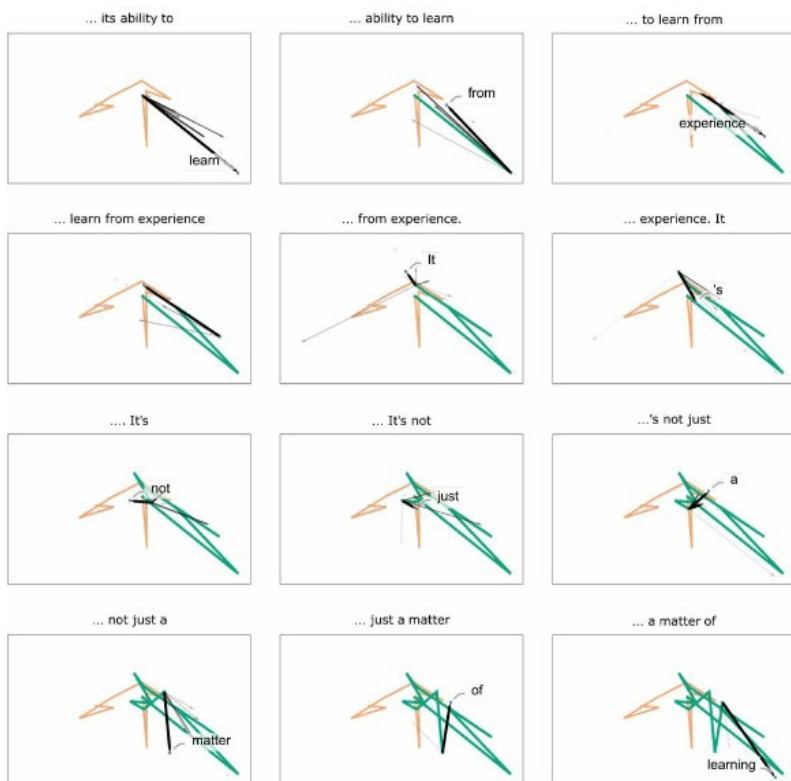


这里显然没有“几何上明显的”运动规律。这一点一点也不奇怪；我们完全期望这是一个更为复杂的故事。例如，即使存在“语义运动规律”，最自然的表述方式（或者说最“变量”的方式）也不明显。

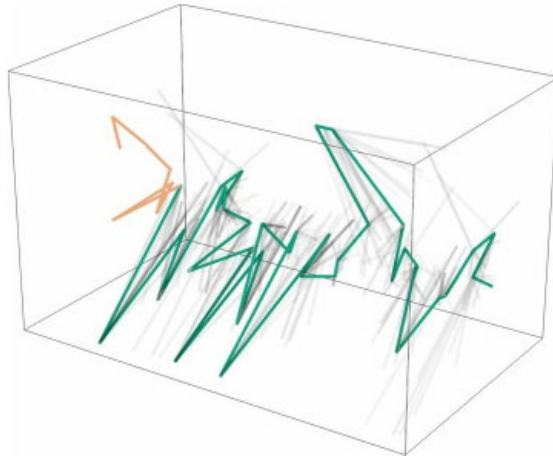
在上面的图片中，我们展示了“轨迹”的几个步骤——在每个步骤中，我们选择 ChatGPT 认为最有可能的单词（“零温度”情况）。但是我们也可以询问在给定点上哪些单词以及以什么概率“可能出现在接下来的位置”：



在这种情况下，我们看到有一个“扇形”，高概率单词似乎在一个或多个明确定义的方向上移动。如果我们进一步移动，会发生什么？这是我们在“沿着”轨迹“移动”时出现的连续“扇形”：



这是一个三维表示，一共进行了 40 步：



是的，这看起来很乱，并没有特别鼓励的想法，即可以通过经验研究“ChatGPT 内部的行为”来识别“类似于数学物理”的“语义运动规律”。但也许我们只是看了“错误的变量”（或坐标系），如果我们只看对的，我们会立即看到 ChatGPT 正在执行类似于地理的运动规律。但是目前，我们还没有准备好从其“内部行为”中“经验性地解码”ChatGPT 已经“发现”关于人类语言“如何组合”的信息。

语义语法和计算语言的力量

要产生“有意义的人类语言”需要什么？过去，我们可能认为只有人类的大脑才能做到。但现在我们知道，ChatGPT 的神经网络也能做到相当不错。不过，也许这就是我们所能达到的极限，没有更简单或更容易理解的方法可行。但我强烈怀疑，ChatGPT 的成功隐含地揭示了一个重要的“科学”事实：实际上，有意义的人类语言比我们所知道的要具有更多的结构和简单性，最终可能会有相当简单的规则来描述如何组成这样的语言。

正如我们上面提到的，句法语法为人类语言中的不同词类提供了规则。但为了处理语义，我们需要更进一步。其中一种方法是不仅考虑语言的句法语法，而且还有语义语法。

对于句法，我们识别名词和动词等词类。但对于语义，我们需要“更细微的区分”。例如，我们可能会识别“运动”这个概念，以及“一个“物体”可以“独立于位置而保持其身份”的概念。这些“语义概念”各有无数具体例子。但对于我们的语义语法，我们只需要一些基本规则，基本上是说“物体”可以“移动”。关于这些可能如何工作有很多话可说（我之前已经说过一些）。但在这里，我只会满足于几点说明，指出一些潜在的前进路径。

值得一提的是，即使一句话在语义语法上是完全正确的，也不意味着它在实践中已经实现（或者可能实现）。例如，“大象去了月球”无疑会“通过”我们的语义语法，但它在我们的实际世界中肯定没有实现过（至少还没有）——但它完全适用于虚构世界。

当我们开始谈论“语义语法”时，我们很快就会问“底层是什么？”它假设了什么“世界模型”？语法语言实际上只是关于从单词构造语言的方法。但语义语法必须涉及某种“世界模型”——一种作为实际单词语言上层的“骨架”的东西。

直到最近，我们可能认为（人类）语言是描述我们“世界模型”的唯一通用方法。几个世纪以前，已经开始对特定类型的事物进行形式化，特别是基于数学。但现在有了一种更一般的形式化方法：计算语言。

是的，这一直是我四十多年来的重大项目（现在体现在 Wolfram 语言中）：开发一个精确的符号表示，可以尽可能广泛地谈论我们关心的事物和抽象概念。因此，我们有城市、分子、图像和神经网络的符号表示，并且我们内置了有关如何计算这些事物的知识。

数十年来，我们已经在很多领域做了大量工作。但过去，我们并没有特别处理“日常话语”。在“I bought two pounds of apples”中，我们可以轻松地表示（并进行营养和其他计算）“两磅苹果”。但我们还没有一个符号表示“我买了”的表示方法（目前还没有）。

这一切都与语义语法的思想相关，以及构建一个通用的符号“构造工具包”来描述概念的目标，从而为我们可以转化为人类语言的“流程”提供规则。

但假设我们有了这个“符号话语语言”，我们会用它做什么呢？我们可以开始生成“局部有意义的文本”。但最终，我们可能希望得到更“全局有意义”的结果，这意味着“计算”更多关于实际世界中可能存在或发生的事情（或者也许是一些一致的虚构世界）。

现在，在 Wolfram Language 中，我们有大量关于许多事物的内置计算知识。但对于一个完整的符号话语语言，我们必须构建关于世界通用事物的其他“演算法”：如果一个物体从 A 到 B，再从 B 到 C，那么它就从 A 到 C 移动了，等等。

在拥有符号话语语言的情况下，我们可以使用它来做出“独立的陈述”。但我们也可以使用它来询问关于世界的问题，类似“Wolfram|Alpha 风格”。或者我们可以用它来陈述我们“想要实现”的事情，可能还要加上一些外部的激活机制。或者我们可以用它来做出断言，也许是关于实际世界，也许是关于我们正在考虑的某个特定世界，无论是虚构的还是其他。

人类语言基本上是不精确的，不仅因为它没有与特定计算实现“捆绑”起来，而且它的含义基本上仅由其用户之间的“社会契约”所定义。但是，计算语言由其本质，具有一定的基本精度，因为最终它所指定的可以“在计算机上无歧义地执行”。人类语言通常可以容忍一定的模糊性（例如，当我们说“行星”时，它是否包括系外行星等），但在计算语言中，我们必须对我们所做的所有区分进行精确而清晰的说明。

在计算语言中，通常利用普通的人类语言来命名，但它们在计算语言中的含义必然是精确的，可能或可能不包含一些典型的人类语言使用中的特定内涵。

如何确定适用于一种普通符号语言的基本“本体论”？这并不容易。这也许是为什么自两千多年前亚里士多德开始时以来，一直没有多少进展。但今天，我们对于如何用计算的方式思考世界已经了解得很多，这些知识对于构建一种符号语言非常有帮助（同时，我们从物理项目和“Ruliad”的思想中获得了一些“基本形而上学”）。

但所有这些在 ChatGPT 的背景下意味着什么呢？通过训练，ChatGPT 有效地“拼凑出”了一定数量的相当令人印象深刻的语义语法。但是，它的成功使我们有理由认为，可以构建更完整的计算语言形式。而且，与我们迄今为止对 ChatGPT 的内部所了解的内容不同，我们可以期望设计计算语言，使其对人类容易理解。

当我们谈论语义语法时，可以将其类比为三段论逻辑。最初，三段论逻辑本质上是一组有关用人类语言表达的陈述的规则。但是当正式逻辑被发展出来时（是的，两千年后），三段论逻辑的最初基本结构现在可以用于构建庞大的“形式塔”，其中包括现代数字电路的操作等等。因此，我们可以预期，更一般的语义语法也将如此。起初，它可能只能处理简单的模式，例如以文本形式表达的模式。但是一旦构建了整个计算语言框架，我们可以预期，它将能够用于建立“广义语义逻辑”的高大塔楼，使我们能够以精确和形式化的方式处理所有种类的事物，这些事物以前从未可见过，除了通过人类语言的“一楼水平”，以及其模糊性。

我们可以将计算语言和语义语法的构建视为一种最终压缩表示事物的方式。因为它使我们能够谈论可能性的本质，而无需涉及存在于普通人类语言中的所有“措辞”。而我们可以将 ChatGPT 的巨大优势视为类似的东西：因为它也在某种程度上“深入到”能够“以语义上有意义的方式组合语言”的程度，而不必考虑不同的可能措辞。

那么，如果我们将 ChatGPT 应用于底层的计算语言会发生什么？计算语言可以描述可能的事物。但是仍然可以添加“流行”的概念，例如基于阅读网络上的所有内容。但是，底层操作计算语言意味着类似 ChatGPT 的系统立即并基本地访问可能不可简化计算的终极工具。这使其成为一个不仅能够“生成合理的文本”，而且可以预期解决关于该文本是否实际上在讨论世界或其它东西的“正确”陈述的所有可能问题的系统。

那么.....ChatGPT 在做什么，它为什么有效呢？

ChatGPT 的基本概念在某种程度上相当简单。首先从网络、图书等人类创造的大量文本样本开始。然后训练神经网络生成“类似于这些”文本。特别是，使其能够从“提示”开始，然后继续生成“类似于已经训练过的”文本。

正如我们所见，ChatGPT 中的实际神经网络由非常简单的元素组成，尽管有数十亿个。神经网络的基本操作也非常简单，基本上是对每个新单词（或部分单词）生成输入时，“将输入通过其元素”（没有任何循环等）。

但值得注意的是，这个过程可以产生文本，它们成功地“类似于”网络、图书等上的内容。它不仅是连贯的人类语言，而且“说出”使用它“阅读”的内容“跟随它的提示”。它并不

总是说出“全局意义上正确的”事情（或者对应于正确的计算）——因为（没有访问 Wolfram|Alpha 的“计算超能力”）它只是根据其训练材料中的内容“听起来正确”的事情。

ChatGPT 的具体工程使其非常引人注目。但是最终（至少在它能够使用外部工具之前），ChatGPT “仅仅”从积累的“常识统计数据”中提取出一些“连贯的文本线索”。但是令人惊奇的是，结果通常非常类似于人类产生的文本。正如我所讨论的，这表明了一些至少在科学上非常重要的东西：人类语言（及其背后的思维模式）在其结构上比我们想象的要简单和更“类似于法律”。ChatGPT 已经暗示了这一点。但是我们可能可以通过语义语法、计算语言等明确地展示它。

ChatGPT 生成文本的过程非常令人印象深刻，结果通常非常类似于我们人类会产生的文本。那么这是否意味着 ChatGPT 像大脑一样工作呢？其底层的人工神经网络结构最终是模拟了大脑的理想化模型。当我们人类生成语言时，许多方面的过程似乎是相似的。

当涉及训练（也称为学习）大脑和当前计算机的不同“硬件”时（以及也许一些未开发的算法思想），ChatGPT 被迫使用一种策略，这种策略可能与大脑相比是相当不同的（在某些方面也更低效）。还有一件事：与典型的算法计算甚至不同，ChatGPT 不会内部“循环”或“重新计算数据”。这不可避免地限制了它的计算能力，即使与当前的计算机相比，但与大脑相比则更是如此。

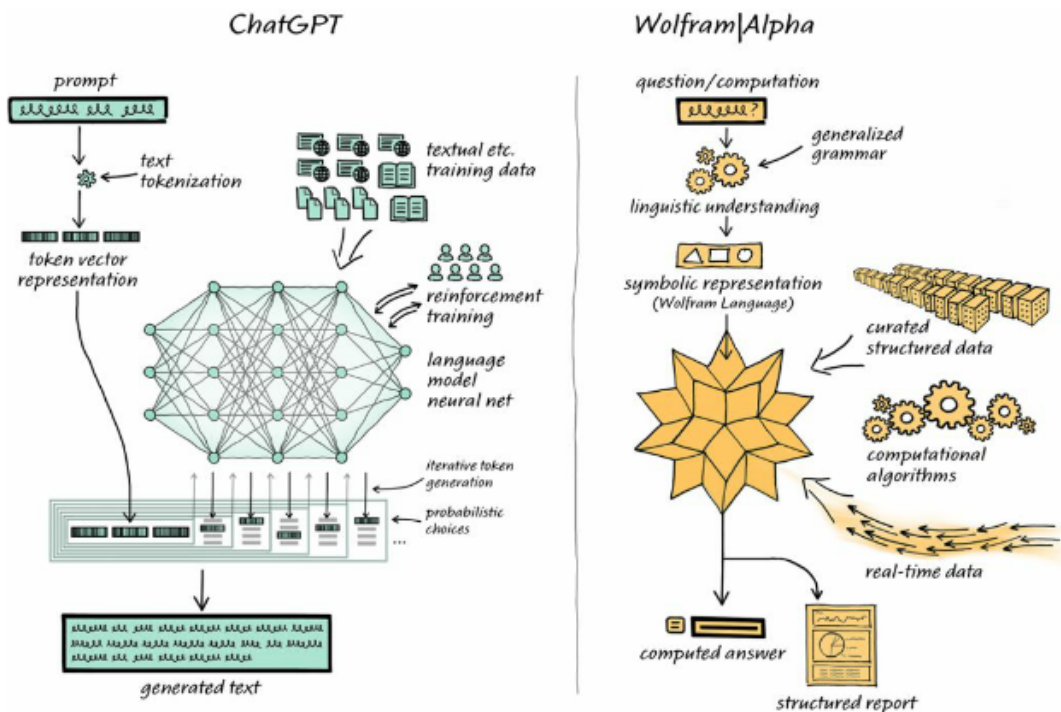
目前还不清楚如何解决这个问题，同时仍能保持对系统的有效训练。但这样做将可能使未来的 ChatGPT 能够做更多“类似大脑的事情”。当然，大脑也有许多做得不好的事情，特别是涉及到不可简化计算的事情。对于这些问题，大脑和像 ChatGPT 这样的系统都需要寻求“外部工具”——如 Wolfram Language。

但现在看到 ChatGPT 已经能够做到的事情令人兴奋。在某种程度上，它是一个很好的例子，说明许多简单的计算元素可以做出令人惊讶和出乎意料的事情。但它也提供了我们在过去两千年中了解人类语言这个核心特征及其背后的思维过程的基本性质和原则的最佳动力。

感谢

我一直在关注神经网络的发展，已经有大约 43 年了，在这段时间里，我与许多人交流过这方面的知识。其中包括：Giulio Alessandrini、Dario Amodè、Etienne Bernard、Taliesin Beynon、Sebastian Bodenstern、Greg Brockman、Jack Cowan、Pedro Domingos、Jesse Galef、Roger Germundsson、Robert Hecht-Nielsen、Geoff Hinton、John Hopfield、Yann LeCun、Jerry Lettvin、Jerome Louradour、Marvin Minsky、Eric Mjolsness、Cayden Pierce、Tomaso Poggio、Matteo Salvarezza、Terry Sejnowski、Oliver Selfridge、Gordon Shaw、Jonas Sjöberg、Ilya Sutskever、Gerry Tesauro 和 Timothee Verdieer。在此篇文章的撰写过程中，我特别感谢 Giulio Alessandrini 和 Brad Klee 的帮助。

Wolfram|Alpha 是将计算知识超能力带给 ChatGPT 的途径



ChatGPT 和 Wolfram|Alpha

当事物突然“奏效”时，总是令人惊叹。这在我们 2009 年推出 Wolfram|Alpha 时发生了。2020 年我们的物理计划也是如此。现在 OpenAI 的 ChatGPT 也正在经历这种情况。

我已经追踪神经网络技术很长时间了（实际上大约 43 年）。即使是过去几年的发展，我也发现 ChatGPT 的表现非常出色。终于有了一个系统，可以成功地生成几乎任何内容的文本，而这些文本与人类写作的非常相似。这令人印象深刻且有用。正如我在其他地方所讨论的那样，我认为它的成功可能在告诉我们一些关于人类思维本质的非常基本的事情。

但是，虽然 ChatGPT 在自动化进行类似人类的主要工作方面取得了显著成就，但并不是所有有用的事情都如此“类人”。有些是更正式和结构化的。实际上，我们文明的伟大成就之一是在过去几个世纪内建立了数学范例、确切科学——最重要的是计算——并创造了一座能力塔，它与纯类人思维可以实现的东西完全不同。

我自己已经深入涉足计算范式数十年，致力于构建一种计算语言，以符号形式尽可能多地表示世界上的事物。在这样做的过程中，我的目标是建立一个系统，可以“计算协助”和增强我和其他人想做的事情。我像人类一样思考问题，但我也可以立即调用 Wolfram Language 和 Wolfram|Alpha，利用一种独特的“计算超级能力”，让我可以做出各种超越人类的事情。

这是一种极为强大的工作方式。而且，重要的是，它不仅对我们人类重要，对人类智能同样重要，如果不更重要——因为它立即赋予了它们一种我们可以称之为计算知识超级能力，借助结构化计算和结构化知识的非人类的力量。

我们刚刚开始探索这对 ChatGPT 意味着什么，但很明显，可以实现很多奇妙的东西。Wolfram|Alpha 和 ChatGPT 以不同的方式做着不同的事情，但它们有一个共同的接口：自然语言。这意味着 ChatGPT 可以像人类一样与 Wolfram|Alpha 进行“交谈”，而 Wolfram|Alpha 将从 ChatGPT 得到的自然语言转换为精确的符号化计算语言，应用其计算知识能力。

几十年来，人工智能的思考方式在“统计方法”和“符号方法”之间存在着二元思维，ChatGPT 使用的就是前者，而 Wolfram|Alpha 则是后者的起点。但是，现在，由于 ChatGPT 的成功以及我们在让 Wolfram|Alpha 理解自然语言方面所做的一切工作，终于有了机会将它们结合起来，创造出比任何一种单独方法都更加强大的东西。

一个基本的例子

在其核心，ChatGPT 是一个生成“跟随模式”的语言输出的系统，这个模式是根据它的训练数据来自于网络、书籍和其他材料的。令人惊奇的是，其输出通常很像人类写的，不仅仅在小尺度上，而是整篇文章都如此。它会表达连贯的思想，用它学过的概念在常常出人意料的有趣方式下展现出来。在语言层面上，它所产生的内容总是“在统计学上合理的”。但是，尽管最终结果令人印象深刻，但这并不意味着它自信地给出的所有事实和计算都是正确的。以下是我刚刚注意到的一个例子（是的，ChatGPT 有内在的内置随机性，所以如果你试一下，你可能不会得到相同的结果）：



“在美国，一英亩土地可以生产两千磅的西蒙斯松树果实，每年可以提供一美元的净收入。”

听起来相当令人信服。但事实证明，这是错的，正如 Wolfram|Alpha 所告诉我们的：



How far is it from Chicago to Tokyo?



Input interpretation

| | | |
|----------|------|-------------------|
| distance | from | Chicago, Illinois |
| | to | Tokyo, Japan |

Result

6313 miles

Unit conversions

10 160 km (kilometers)

公里等于英里的数量是不正确的。实际上，这个数字的正确值是 0.621371，你可以用 Wolfram|Alpha 或其他工具来验证它。当然，这恰恰是 Wolfram|Alpha 擅长的：将其转化为精确计算，并根据它的结构化和精心策划的知识进行计算。但有趣的是，我们可以考虑 Wolfram|Alpha 在这方面自动帮助 ChatGPT。可以使用编程方式询问 Wolfram|Alpha 这个问题（也可以使用 Web API 等）：

```
In[ ]:= WolframAlpha["what is the distance from Chicago to Tokyo", "SpokenResult"]
```

```
Out[ ]:= The distance from the center of Chicago, Illinois to the center of Tokyo is about 6313 miles
```

现在，再向 ChatGPT 询问这个问题，并附上这个结果：



How far is it from Chicago to Tokyo? The distance from the center of Chicago, Illinois to the center of Tokyo is about 6313 miles



Thank you for correcting me. You are correct that the distance from the center of Chicago, Illinois to the center of Tokyo, Japan is about 6313 miles (10160 kilometers). This distance can vary slightly depending on the specific location within each city that is being used as the starting and ending point.



How far is it from Chicago to Tokyo?



The distance from Chicago, Illinois to Tokyo, Japan is approximately 6313 miles (10160 kilometers). This is a very long distance and it would take a significant amount of time to travel from one place to the other by any mode of transportation. The actual flight time from Chicago, IL to Tokyo, Japan is about 16 hours, but this does not take into account factors such as wind and the layover time at the airport.

ChatGPT 非常客气地接受了更正，如果您再次询问这个问题，它会给出正确答案。显然，处理与 Wolfram|Alpha 的交互可能有更简洁的方法，但很高兴看到即使这种非常简单的纯自然语言方法基本上已经有效。

但为什么 ChatGPT 首先会犯这个错误呢？如果它在训练中某个地方（例如来自 Web）看到了芝加哥和东京之间的特定距离，它当然可以做对。但这是一种情况，神经网络容易从许多城市之间的距离示例中进行推广，但是这样不足以做到正确；实际上需要一种实际的计算算法。Wolfram|Alpha 处理事物的方式非常不同。它采用自然语言，然后——假设可能性——将其转换为精确的计算语言（即 Wolfram 语言），在这种情况下：

```
In[ ]:= GeoDistance[ Chicago CITY , Tokyo CITY ]  
Out[ ]:= 6296.06 mi
```

城市的坐标和计算它们之间距离的算法是 Wolfram 语言内置的计算知识的一部分。是的，Wolfram 语言具有大量内置的计算知识——这是我们几十年来的工作成果，精心策划的内容现在是一个庞大的不断更新的数据集，实施（并经常发明）方法、模型和算法，并系统地为一件事建立整个连贯的计算语言。

更多例子

ChatGPT 和 Wolfram|Alpha 的工作方式非常不同，具有非常不同的优势。但为了理解 ChatGPT 可以利用 Wolfram|Alpha 的优势的地方，让我们讨论一些 ChatGPT 单独不能完全正确处理的情况。ChatGPT 和人类一样，在数学方面经常难以应对。



What is 3 to the power 73?



3 to the power 73 is equal to 3^{73} , which is approximately 14,091,714,236.

To calculate an exponent, you can use the formula "base"^{exponent} = "result". For example, to calculate 3^{73} , you would plug in 3 for the base and 73 for the exponent to get 3^{73} = approximately 14,091,714,236.

It is important to note that this result is just an estimate, as the actual value of 3^{73} is very large and may not be accurately represented by most calculators or computers.

这是一个有趣的论文式回答。但实际结果是错误的：



What is 3 to the power 73?



Input

3^{73}

Result

67 585 198 634 817 523 235 520 443 624 317 923



What is 3 to the power 73?

Scientific notation

$6.7585198634817523235520443624317923 \times 10^{34}$

Number names

Full name

67 decillion ...

67 million billion billion billion ...

但是如果 ChatGPT “请教” Wolfram|Alpha，它当然能够得到正确答案。让我们尝试一些稍微复杂的例子：



circumference of an ellipse with half axes 3 and 12



The circumference of an ellipse can be approximated using the formula:

$$\text{Circumference} \approx \pi * \sqrt{(a^2 + b^2) / 2}$$

where a and b are the lengths of the semi-major and semi-minor axes of the ellipse, respectively. The semi-major axis is the longer of the two axes that pass through the center of the ellipse and divide it into two halves, and the semi-minor axis is the shorter of the two axes.

In an ellipse with semi-major axis 3 and semi-minor axis 12, the circumference would be:

$$\begin{aligned} \text{Circumference} &\approx \pi * \sqrt{(3^2 + 12^2) / 2} \\ &\approx \text{approximately } 27.3 \end{aligned}$$

The circumference of this ellipse is approximately 27.3.

It is important to note that this formula is an approximation, and the actual circumference of an ellipse can be calculated more accurately using more complex formulas.

乍一看，这个结果看起来很好，我倾向于相信它。但事实证明它是错误的，Wolfram|Alpha 可以告诉我们：



circumference of an ellipse with half axes 3 and 12



Input interpretation

ellipse

semi-axis lengths 3 | 12

circumference

Result

$$48 E\left(\frac{15}{16}\right) \approx 51.4705$$

$E(m)$ is the complete elliptic integral of the second kind with parameter $m = k^2$

是的，在没有能够请教 Wolfram|Alpha 的情况下，用 ChatGPT 完成数学作业可能是不明智的。它可以给出一个非常合理的答案：



What is the integral of $x^2 \cos(2x)$?



The integral of $x^2 \cos(2x)$ is a mathematical expression representing the area under the curve of the function $y = x^2 \cos(2x)$. It can be calculated using the following formula:

$$\int x^2 \cos(2x) dx = \frac{1}{4} x^3 \sin(2x) - \frac{1}{8} x \cos(2x) + C$$

where C is an arbitrary constant and \int represents the integral symbol.

The integral of a function represents the area under the curve of the function between two points. It can be calculated by finding the area of the region bounded by the curve, the x -axis, and the vertical lines representing the two points.

The formula for the integral of $x^2 \cos(2x)$ can be derived using integration techniques and rules. It is important to note that this formula is valid for certain ranges of x and may not give an accurate result for other values of x .

有很多情况下，“不真正理解事物的含义”可能会导致麻烦：

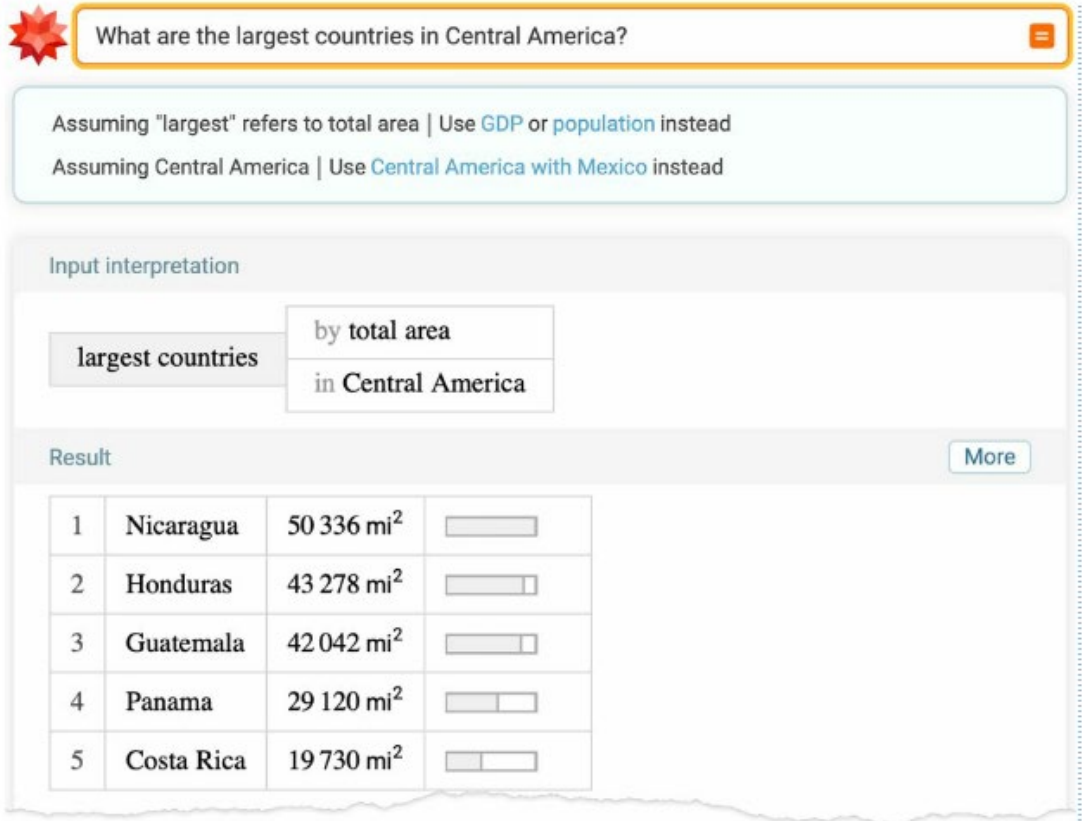


What's the second largest country in central america



The second largest country in Central America (by land area) is Guatemala. Guatemala is located in the northern part of Central America, bordered by Mexico to the north and west, Belize and the Caribbean Sea to the east, and Honduras and El Salvador to the south. With a land area of approximately 108,889 square kilometers (42,042 square miles), Guatemala is slightly smaller than the largest country in Central America, which is Nicaragua.

这听起来很有说服力，但它是不正确的：



What are the largest countries in Central America?

Assuming "largest" refers to total area | Use [GDP](#) or [population](#) instead
Assuming Central America | Use [Central America with Mexico](#) instead

Input interpretation

largest countries by total area in Central America

Result [More](#)

| | | | |
|---|------------|------------------------|----------------------|
| 1 | Nicaragua | 50 336 mi ² | <input type="text"/> |
| 2 | Honduras | 43 278 mi ² | <input type="text"/> |
| 3 | Guatemala | 42 042 mi ² | <input type="text"/> |
| 4 | Panama | 29 120 mi ² | <input type="text"/> |
| 5 | Costa Rica | 19 730 mi ² | <input type="text"/> |

ChatGPT 似乎已经在某个地方正确地学习了这个基础数据，但是它“不理解它的含义”以至于不能正确地对数字进行排序：



what are the largest countries in central america

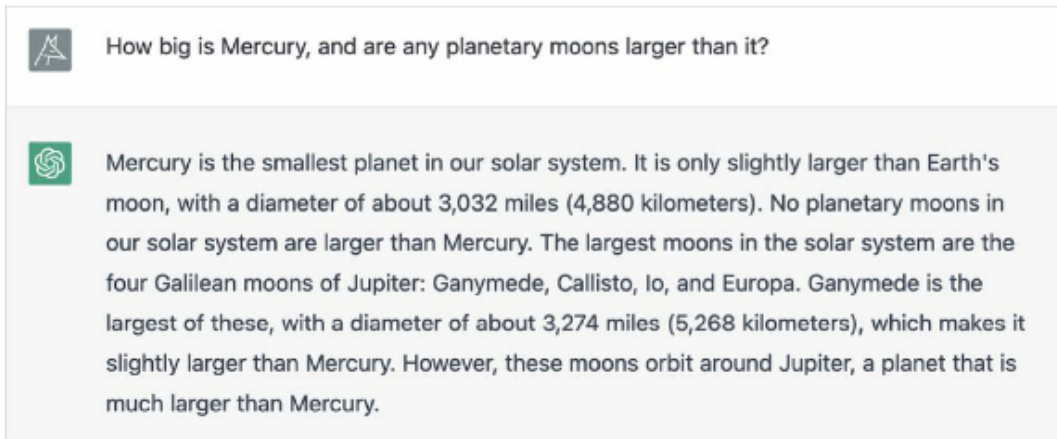
The largest countries in Central America, ranked by land area, are:

1. Nicaragua - 130,370 square kilometers (50,193 square miles)
2. Guatemala - 108,889 square kilometers (42,042 square miles)
3. Honduras - 112,492 square kilometers (43,278 square miles)
4. El Salvador - 21,041 square kilometers (8,124 square miles)
5. Costa Rica - 51,100 square kilometers (19,656 square miles)
6. Panama - 78,200 square kilometers (30,193 square miles)

It's important to note that these figures represent the total land area of each country, including any islands that are part of the country. Some of the countries in Central America, such as Honduras and El Salvador, are relatively small in size, while others, such as Nicaragua and Guatemala, are larger.

是的，我们可以想象找到一种方法来“修复这个特定的漏洞”。但是，重要的是要意识到，基于生成语言的 AI 系统（如 ChatGPT）的基本思想在处理有结构的计算问题时并不是一个好的选择。换句话说，即使修补了几乎无限数量的“漏洞”，也很难达到 Wolfram|Alpha 在其结构化方式上所能实现的几乎无限小的一部分。

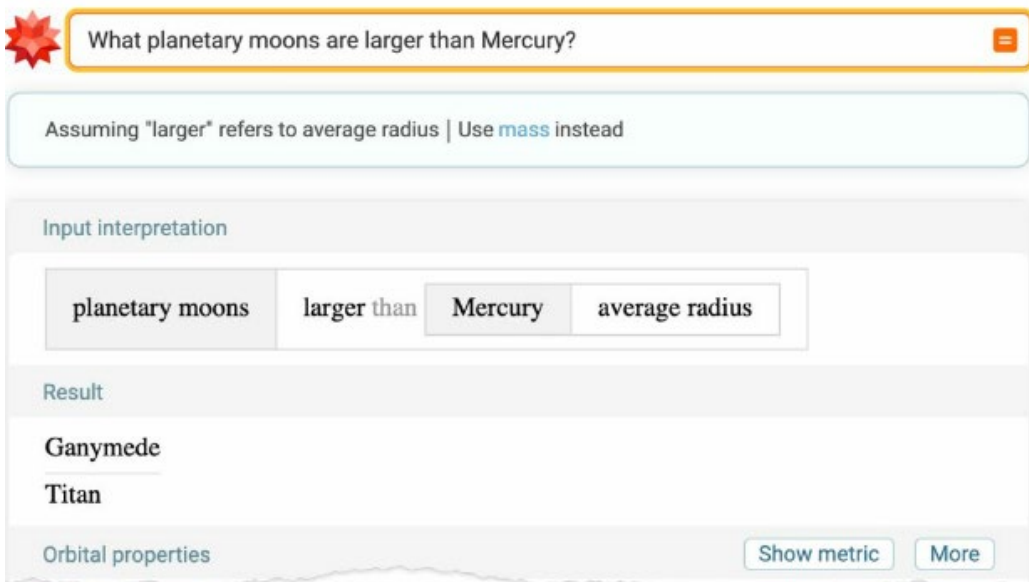
而且，计算链条越复杂，就越有可能需要调用 Wolfram|Alpha 才能得到正确的答案。在这里，ChatGPT 给出了一个相当混乱的答案：



How big is Mercury, and are any planetary moons larger than it?

Mercury is the smallest planet in our solar system. It is only slightly larger than Earth's moon, with a diameter of about 3,032 miles (4,880 kilometers). No planetary moons in our solar system are larger than Mercury. The largest moons in the solar system are the four Galilean moons of Jupiter: Ganymede, Callisto, Io, and Europa. Ganymede is the largest of these, with a diameter of about 3,274 miles (5,268 kilometers), which makes it slightly larger than Mercury. However, these moons orbit around Jupiter, a planet that is much larger than Mercury.

而且，正如 Wolfram|Alpha 告诉我们的那样，它的结论是不正确的（因为 Wolfram|Alpha 在某种程度上已经“知道”）：



What planetary moons are larger than Mercury?

Assuming "larger" refers to average radius | Use mass instead

Input interpretation

| | | | |
|-----------------|-------------|---------|----------------|
| planetary moons | larger than | Mercury | average radius |
|-----------------|-------------|---------|----------------|

Result

- Ganymede
- Titan

Orbital properties [Show metric](#) [More](#)

每当涉及具体的（例如定量）数据，即使在相当原始的形式下，事情往往也更多地成为一个“Wolfram|Alpha 的故事”。以下是一个例子，受到了 Wolfram|Alpha 一直以来最受欢迎的测试查询“土耳其有多少只火鸡？”的启发：



Tell me about livestock populations in Turkey



Turkey is a major producer of livestock, including poultry, cattle, sheep, and goats. According to data from the Food and Agricultural Organization of the United Nations (FAO), as of 2020, Turkey had the following livestock populations:

- Poultry: Turkey has a large poultry industry, with a population of approximately 537 million chickens, 37 million turkeys, and 3 million ducks.
- Cattle: Turkey has a relatively small cattle population, with approximately 7.4 million head of cattle.
- Sheep: Turkey has a larger sheep population, with approximately 25.6 million sheep.
- Goats: Turkey also has a significant goat population, with approximately 10.7 million goats.

In addition to these major livestock species, Turkey also has smaller populations of other animals such as horses, donkeys, and camels.

这个回答一开始似乎是完全可以信的，它甚至引用了一个相关的来源。但实际上，这个数据基本上是“虚构”的：



Livestock populations in Turkey



Input interpretation

Turkey livestock population

Result

More

| | |
|----------|------------------------------|
| cattle | 18 million (2020 estimate) |
| chickens | 379 million (2020 estimate) |
| goats | 12 million (2020 estimate) |
| pigs | 990 (2020 estimate) |
| sheep | 42.1 million (2020 estimate) |

不过，非常好的是，ChatGPT 可以轻松地被制作成“要求事实检查”的形式：



Give me a simple keyword query that could be answered by each numerical value in your response; only list the queries



- "Number of chickens in Turkey"
- "Number of turkeys in Turkey"
- "Number of ducks in Turkey"
- "Number of cattle in Turkey"
- "Number of sheep in Turkey"
- "Number of goats in Turkey"

现在通过 Wolfram|Alpha API 将它们馈送进去：

```
In[ ]:= WolframAlpha[#, "SpokenResult"] & /@  
{"Number of chickens in Turkey", "Number of turkeys in Turkey",  
"Number of ducks in Turkey", "Number of cattle in Turkey",  
"Number of sheep in Turkey", "Number of goats in Turkey"}
```

```
Out[ ]:= {In 2020, Turkey had about 379 million chickens as livestock,  
In 2020, Turkey had about 4.8 million turkeys as livestock,  
In 2020, Turkey had 560000 ducks as livestock,  
In 2020, Turkey had about 18 million cattle as livestock,  
In 2020, Turkey had about 42.1 million sheep as livestock,  
In 2020, Turkey had about 12 million goats as livestock}
```

现在我们可以要求 ChatGPT 修正其最初的响应，注入这些数据（甚至在加粗处显示它所做的修改）：



Insert this corrected data into your original response, surrounding each modified value with "***":

```
{"In 2020, Turkey had about 379 million chickens as livestock", "In 2020, Turkey had about 4.8 million turkeys as livestock", "In 2020, Turkey had 560000 ducks as livestock", "In 2020, Turkey had about 18 million cattle as livestock", "In 2020, Turkey had about 42.1 million sheep as livestock", "In 2020, Turkey had about 12 million goats as livestock"}
```



Here is the corrected data inserted into my previous response, with the modified values surrounded by "***":

Turkey is a major producer of livestock, including poultry, cattle, sheep, and goats. According to data from the Food and Agricultural Organization of the United Nations (FAO), as of 2020, Turkey had the following livestock populations:

- Poultry: **In 2020, Turkey had about 379 million chickens as livestock and 4.8 million turkeys as livestock**, as well as about **560000 ducks**.
- Cattle: **In 2020, Turkey had about 18 million head of cattle**.
- Sheep: **In 2020, Turkey had about 42.1 million sheep**.
- Goats: **In 2020, Turkey had about 12 million goats**.

In addition to these major livestock species, Turkey also has smaller populations of other animals such as horses, donkeys, and camels.

当涉及实时数据（或与位置等有关的数据）或计算时，“注入事实”的能力特别有用。ChatGPT 不会立即回答这个问题：



What planets can I see tonight?



I'm sorry, but I don't have any current information on what planets are visible in the sky tonight. In order to find out what planets are visible tonight, you'll need to use a planetarium app or website. These tools use your location and the current time to provide information on which planets are visible in the sky. Some popular planetarium apps and websites include Sky Guide, Star Walk, and Stellarium. Alternatively, you can check a printed sky map or ask an astronomer for information on what planets are visible from your location.

但这里有一些相关的 Wolfram|Alpha API 输出：

```
In[ ]:= Values[WolframAlpha["What planets can I see tonight",  
{"Input", "PropertyRanking:PlanetData"}, "Plaintext"]]
```

```
Out[ ]:= {planets | above the horizon | 10:00 pm, Mercury | Mercury is below the horizon  
Venus | Venus is below the horizon  
Mars | Mars is above the horizon  
Jupiter | Jupiter is above the horizon  
Saturn | Saturn is below the horizon  
Uranus | Uranus is above the horizon  
Neptune | Neptune is above the horizon}
```

如果我们将这个问题输入 ChatGPT，它会生成一个漂亮的“文章风格”的结果：



Revise your answer, incorporating this data:

```
{"planets | above the horizon | 10:00 pm",  
"Mercury | Mercury is below the horizon  
Venus | Venus is below the horizon  
Mars | Mars is above the horizon  
Jupiter | Jupiter is above the horizon  
Saturn | Saturn is below the horizon
```



According to the information provided, at 10:00 PM, the planets that are above the horizon are Mars, Jupiter, Uranus, and Neptune. Mercury and Venus are below the horizon, and Saturn is also below the horizon. I should note that the visibility of planets can vary depending on your location and the time of year. I recommend using a planetarium app or website to get the most accurate and up-to-date information on which planets are visible from your location.

有时，在计算和人类之间存在有趣的相互作用。这里是一个相当古怪的问题问 Wolfram|Alpha（它甚至会检查你是否想要“软冰淇淋”）：



How many calories are there in a cubic light year of ice cream?



Assuming ice cream | Use [soft-serve ice cream](#) instead

Assuming any type of ice cream | Use [ice cream, vanilla](#) or [more](#) instead

Input interpretation

ice cream

amount

1 ly³ (cubic light year)

total calories

Average result

Show details

1.4×10^{54} Cal (dietary Calories)

Unit conversions

5.8×10^{57} J (joules)

5.8×10^{64} ergs
(unit officially deprecated)

ChatGPT 刚开始对于容积的概念有点困惑：



How many calories are in a cubic light year of ice cream?



I'm sorry, but it is not possible to have a cubic light year of anything, let alone ice cream. A light year is a unit of distance, not volume, and is equal to approximately 5.88 trillion miles. It is a measure of the distance that light can travel in one year. Therefore, it is not possible to have a cubic light year of any substance, including ice cream.

但接着它好像“意识到”那么多的冰淇淋有点荒谬：



A cubic light year is an extremely large unit of volume. In fact, it is so large that it is not possible to fit any known substance, including ice cream, into a volume of this size.

For reference, a light year is a unit of distance that is equal to the distance that light travels in one year. The distance that light travels in one year is approximately 5.88 trillion miles (9.46 trillion kilometers).

Given the size of a light year, it is clear that a cubic light year is an impossibly large unit of volume. Therefore, it is not possible to calculate the number of calories in a cubic light year of ice cream, as there is simply no way to fit that much ice cream into a single volume.

未来之路

机器学习是一种强大的方法，特别是在过去的十年中，它取得了一些非凡的成就，其中 ChatGPT 就是最新的例子。图像识别，语音转文本，语言翻译，在每个这样的领域里，阈值被突破了，通常是相当突然的。某些任务从“基本不可能”变成了“基本可行”。

但结果从根本上来说都不是“完美的”。也许某些任务在 95% 的时间内工作得很好，但无论如何，剩下的 5% 都是难以捉摸的。对于某些目的而言，这可能被认为是一个失败。但关键点是，通常有各种重要的用例，对于这些用例而言，95% 已经足够好了。也许是因为输出的结果本身并没有一个“正确答案”。也许是因为只是试图揭示人类或系统算法将从中挑选或改进的可能性。

一个可以一次产生一个标记的文本的几百亿参数的神经网络能够做到 ChatGPT 所做的事情，这完全是非凡的。鉴于这种戏剧性而意外的成功，人们可能会认为，如果能够“训练足够大的网络”，就可以用它来做任何事情。但事实并非如此。关于计算的基本事实，尤其是计算的不可简化性概念，表明它最终无法做到。但更重要的是，我们在机器学习的实际历史中看到的。将会有巨大的突破（如 ChatGPT）。改进不会停止。但更重要的是，会发现使用情况，这些情况成功地完成了可以做的事情，并且不会被不能做到的事情所阻挡。

是的，还有很多情况，其中“原始 ChatGPT”可以帮助人们的写作，提供建议或生成对各种文档或交互有用的文本。但当涉及到必须完美完成的事情时，机器学习并不是解决问题的方式——人类也是如此。

以上的例子正好展示了这一点。ChatGPT 在“人类般的部分”表现出色，那些没有明确的“正确答案”的部分。但是当它在某些明确的问题上“置于聚光灯下”时，它经常失灵。但整个重点在于，有一种很好的方法可以解决这个问题——通过将 ChatGPT 连接到 Wolfram|Alpha 和其所有的计算知识“超能力”上。

在 Wolfram|Alpha 内部，一切都被转化为计算语言和精确的 Wolfram 语言代码，这在某种程度上必须是“完美的”，才能可靠地使用。但关键的是，ChatGPT 不必生成这个，它可以生成它通常的自然语言，然后 Wolfram|Alpha 可以利用其自然语言理解能力将自然语言转化为精确的 Wolfram 语言。

在许多方面，可以说 ChatGPT 从来没有真正“理解”过事物，它只是“知道如何产生有用的东西”。但对于 Wolfram|Alpha 来说，情况则不同。因为一旦 Wolfram|Alpha 将某些东西转换为 Wolfram 语言，它拥有的是一个完整、精确、正式表示，可以可靠地计算事物。不用说，我们有许多“人类感兴趣”的事物，我们没有正式的计算表示——尽管我们仍然可以用自然语言谈论它们，但可能不够准确。对于这些事物，ChatGPT 是独立的，具有其非常出色的能力。

但就像我们人类一样，有时候 ChatGPT 需要更正式和精确的“动力助推”。但关键是，在表达需求时它不必是“正式和精确”的。因为 Wolfram|Alpha 可以用它的母语——自然语言与之交流。而 Wolfram|Alpha 在转换成自己的母语——Wolfram 语言时会负责“添加正式性和精确性”。我认为这是一个非常好的情况，有着巨大的实用潜力。

而且这个潜力不仅限于典型的聊天机器人或文本生成应用程序的层面，还涉及到数据科学或其他形式的计算工作（或编程）。从某种意义上说，这是一个立即获得最佳状态的方法：ChatGPT 的人类化世界与 Wolfram 语言的计算精确世界相结合。

那么 ChatGPT 直接学习 Wolfram 语言呢？好吧，是的，它可以这样做，实际上它已经开始了。最终，我完全期望像 ChatGPT 这样的东西将能够直接在 Wolfram 语言中运行，并在此过程中具有很强的能力。这是一个有趣而独特的情况，它是由 Wolfram 语言的特点作为一个可以广泛谈论世界上事物和计算方面的语言而实现的。

Wolfram 语言的整个概念是将我们人类思考的事物转化为可计算的表示并与之一起工作。普通的编程语言旨在提供一种方法来明确告诉计算机要做什么。Wolfram 语言——在其作为全面计算语言的角色中——是关于比这更大的事情。实际上，它旨在成为一种人类和计算机都可以“计算思考”的语言。

许多世纪前，数学符号的发明首次提供了一种流畅的媒介，用于对事物进行“数学思考”。它的发明很快导致了代数、微积分和最终的各种数学科学的发展。Wolfram 语言的目标是为计算思维做类似的事情，现在不仅仅是为人类，还可以通过计算范式开启所有“计算 X”领域。

我个人受益匪浅，因为 Wolfram 语言是一种“思考语言”，而且在过去的几十年里，通过人们通过 Wolfram 语言的媒介“以计算的方式思考”，已经取得了很大进展，这太棒了。那么 ChatGPT 怎么样呢？好吧，它也可以参与进来。我还不确定它将如何运作。但这不是关于 ChatGPT 学习如何进行 Wolfram 语言已经知道如何进行的计算。这是关于 ChatGPT 学习如何像人类一样使用 Wolfram 语言。这是关于 ChatGPT 提出“创造性论文”的类比，但现在不是用自然语言而是用计算语言编写。

我长期讨论过由人类编写的计算论文的概念-以混合自然语言和计算语言进行交流。现在的问题是 ChatGPT 能否编写这些论文，并能够使用 Wolfram 语言作为一种交付“有意义的交流”的方式，不仅仅是向人类，而且还向计算机。是的，实际执行 Wolfram 语言代码涉及到一个可能有趣的反馈环路。但是，关键的一点是，Wolfram 语言代码所代表的“想法”的丰富性和流动性——与普通编程语言不同——更接近于 ChatGPT 在自然语言中“神奇地”处理的东西的类型。

换句话说，Wolfram 语言，就像自然语言一样，足够表达意思，我们可以想象用它写一个有意义的 ChatGPT 提示。是的，Wolfram 语言可以直接在计算机上执行，但作为 ChatGPT 提示，它可以用来“表达一个想法”，其“故事”可以继续。它可能描述某些计算结构，让 ChatGPT“即兴发挥”，说出有关该结构的计算话题，这样可以根据 ChatGPT 通过阅读许多人类写作学习到的内容，从而“吸引人类的兴趣”。

ChatGPT 的意外成功打开了各种令人兴奋的可能性。但目前有一个即时机会，那就是通过 Wolfram|Alpha 赋予 ChatGPT 计算知识的超能力，这样它不仅可以生成“可信的人类般输出”，而且还可以利用 Wolfram|Alpha 和 Wolfram 语言所囊括的整个计算和知识领域。

Additional Resources

“What Is ChatGPT Doing ... and Why Does It Work?”

Online version with runnable code

wolfr.am/SW-ChatGPT

“Machine Learning for Middle Schoolers” (by Stephen Wolfram)

A short introduction to the basic concepts of machine learning

wolfr.am/ML-for-middle-schoolers

Introduction to Machine Learning (by Etienne Bernard)

A book-length guide to modern machine learning with runnable code

printed: wolfr.am/IML-book ; online wolfr.am/IML

Wolfram Machine Learning

Machine learning capabilities in the Wolfram Language

wolfr.am/core-ML

Machine Learning at Wolfram U

Interactive classes and courses on machine learning at a variety of levels

wolfr.am/ML-courses

“How Should We Talk to AIs?” (by Stephen Wolfram)

A short essay from 2015 on communicating with AIs in natural and computational language

wolfr.am/talk-AI

The Wolfram Language wolfram.com/language

Wolfram|Alpha wolframalpha.com

Online link to all resources:

wolfr.am/ChatGPT-resources