李宏毅生成式人工智能教程
LeeGenAI Tutorial

https://github.com/datawhalechina/leegenai-tutorial

Qi Wang　Yiyuan Yang　Ji Jiang　编著

版本：0.2.0

2024 年 6 月 17 日

前言

生成式人工智能是最近非常火的话题，也是目前人工智能着重发展的几个方向之一。李宏毅老师是台湾大学的教授，李老师的课程简单清晰、通俗易懂，适合初学者入门。本教程主要内容源于李宏毅老师的课程《生成式人工智能导论》（2024 春），并在其基础上进行了一定的原创。

致谢

特别感谢 Sm1les、LSGOMYP对本项目的帮助与支持。

扫描下方二维码，然后回复关键词“**李宏毅深度学习**”，即可加入“LeeDL/GenAI-Tutorial 读者交流群”

Datawhale
一个专注于 AI 领域的开源组织

目录

第 1 章　生成式人工智能

1.1　生成式人工智能基本概念

在讲生成式人工智能之前，先介绍下什么是人工智能（Artificial Intelligence，AI）。人工智能，顾名思义，就是人所创造出来的智能、机器所展现的智能，不是人本身的智能。但是智能又是什么，每个人心里所想的智能通通都是不一样的，有人觉得 ChatGPT 是一种人工智能；有人觉得 ChatGPT 不能称为人工智能，要一个机器人跑来跑去才算是人工智能；有人觉得电脑要会选土豆才是人工智能。因此人工智没有一个标准的定义，每个人心里所想的人工智能都是不一样的。人工智能相关的论文里面几乎不会提人工智能这个词，因为它是一个没有清楚定义的词汇。人工智能是我们想要达到的一种模糊的目标，而不是一个特定的技术。

相比之下，生成式人工智能（Generative AI，GAI）的定义就比较明确了。GAI 的目的是让机器产生**复杂且有结构**的结果。比如文章，它是由一连串的文字所构成的；比如图像，它是由一组像素所组成的；比如语音，它是由一系列取样点所组成的。

所谓“复杂”到底应该要复杂到什么程度？答案是要复杂到没有办法穷举。例如，要写一篇 100 字的中文文章，实际上就是要将 100 个字连接成有意义的语句。假设我们掌握的文字共有 1000 个（当然，实际上中文常用字不止 1000 个），那么 100 字的文章就有 $1000^{100} = 10^{300}$ 种可能性。我们认为 10^{300} 种可能性是没有办法穷举的，因为宇宙中原子的数量估计只有 10^{80} 而已。这就是复杂。所谓的“有结构”又是什么意思？答案是生成式 AI 生成的结果是有结构的对象，比如序列、列表、树等。

我们可以反过来，说明什么样的问题不是 GAI 的问题，来更了解什么是 GAI。

分类（classification）问题就不是 GAI 的问题。分类问题是要让机器从有限的选项中去做选择，我们已经告诉机器说有哪几个选项是可以选的。例如，邮箱有垃圾邮件检测的功能，有时候如果检测到一封信是垃圾邮件，会自动放到垃圾邮件夹里面。垃圾邮件检测系统的输出只有两个选项：邮件是垃圾邮件、邮件不是垃圾邮件，它不是 GAI。再例如，人脸识别系统看起来数据量很大，但仍然只是分类问题，因为它只是在有限的选项中做选择。

我们可以做一个图像识别系统，这个图像识别系统只能够分类一张肯定包括猫或狗的图片里面是有猫还是有狗，它只会说有猫或有狗，只有两个可能的答案，从有限的选项中做选择，这不是生成式人工智能。

> 当机器从有限的选项中做选择的时候，这不是一个生成式 AI 的问题。

AI 的目标是虚无缥缈的，每个人对此的想像都不太一样，而 GAI 的目标则是其中比较清晰的一个：让机器可以产生复杂且有结构的结果。

另外一个常常跟 AI 一起提到的概念——机器学习（Machine Learning，ML），它的目标是让机器从数据里面找出一个函数。举个例子，比如有一个函数，它输出为 y，输入为 x，我们用 $f(x)$ 来表示它。我们知道它的形式为

$$y = f(x) = ax + b \tag{1.1}$$

现在，我们知道输入 $x = 4$ 时，它的输出是 $y = 5$；输入 $x = 2$ 时，它的输出是输出 $y = -1$，那么 a 和 b 应该分别是多少呢？我们可以解出 $a = 3, b = -7$。知道了 a 和 b 的值后，我们就可以代入新的 x，算出新的 y。例如 $x = 1$，则 $y = 1 \times 3 - 7 = -4$。

整个问题对于我们来说非常简单，是中学阶段就可以学到的解方程组问题。但如果这是一个机器学习问题，则需要让机器通过一系列的方法，自动找到参数。对于我们的问题，就是去自动找到 a 和 b。

> a, b 这两个未知数，在机器学习的领域里面，又称为参数（parameter）。所以本书提到参数的时候，就是指要找出的未知数。

为什么我们不用人力解方程组，而是要让机器自动找参数呢？因为现实中要面对的问题，往往比 $f(x) = a + b$ 还要复杂的多。

假设今天要让机器学会分辨一张图片中是猫还是狗，我们就需要找到这样一个函数 f，它的输入是一张图片，输出只有两个可能：猫、狗。如果可以找到这个函数，你就可以拿它来识别图片中是猫还是狗。

这个函数 f 显然非常复杂。f 如果只采用 $ax + b$ 这样的形式，显然不可能可以分辨猫和狗。f 里面可能有上万个参数，比如 $a, b, c, d, \ldots$ 等，如图 1.1 所示。

> 本书中，模型指的是一个带有大量未知参数的函数。

图 1.1　识别猫狗的函数

图 1.2　模型的输入与输出

f 有上万个参数，我们都没有办法想像这个函数应该长什么样子。但机器学习这个技术，可以帮助我们在给定一些输入输出的条件限制的情况下，把这个函数找出来。例如，如图 1.2 所示，我们为机器输入一组数据，其中有带有猫的一批图片，并标注了它们是猫；也有带有狗的一批图片，并且标注了它们是狗。机器在接受足够多这样的信息后，就能自动找到这上万的参数，从而找到函数 f。

机器学习的“学习”二字，指的就是把这个带有上万个参数的函数找出来的过程，这个过程又称为训练（training）；我们为机器提供的数据是能够帮助我们找出上万个参数的输入输出的条件限制，这种数据就叫做训练数据；找到这个函数长什么样子之后，我们就可以为模型提供一张它没有见过的新的图片，看看机器能否给出正确的输出，这件事情叫做测试（testing），又称为推断（inference）。

而在机器学习领域，通常如何表示这个有上万个参数的模型呢，今天往往它被表示成一个神经网络（neural network），其结构如图1.3所示，关于神经网络的具体细节本书后面会再介绍，目前我们只需要了解这么多。

其实，神经网络本质上还是一个有非常多参数的函数，只不过它们被组织成网状的形式。当我们将带有大量参数的函数表示成一个神经网络，然后让机器自动把这些参数找出来，这种技术就叫做深度学习（Deep Learning，DL），所以深度学习是机器学习的一种。

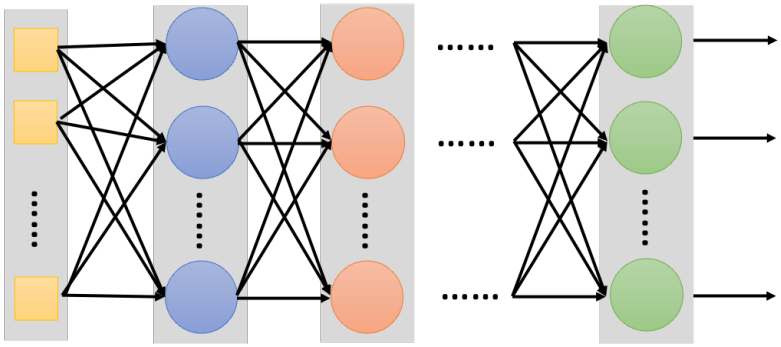


图 1.3　神经网络

机器学习是一种手段，生成式人工智能可以用机器学习来解，但我们也可以用非机器学习的方法来解。机器学习不是只能解生成式人工智能，也可以解其他的问题，比如说分类的问题，而深度学习是机器学习的一种。生成式人工智能、深度学习、机器学习的关系如图 1.4 所示。如果把生成式人工智能放在机器学习里面，也可以接受，因为生成式人工智能是一个非常困难的问题，也许其他手段都不足以达成让你满意的结果，所以如今的生成式人工智能通常都是以深度学习技术来达成的。如果看网络的文章，在描述机器学习、深度学习、生成式人工智能关系的时候，往往会说生成式人工智能是深度学习的一种，如图 1.5 所示。虽然深度学习是一个技术，生成式人工智能是一个目标，但是现在通常会用机器学习的，通常会用深度学习的技术来达成生成式人工智能，所以这种说法也能接受。

1.2　生成式人工智能的原理

1.2.1　ChatGPT 的原理

到底深度学习、机器学习要来怎么解生成式人工智能的问题呢？ChatGPT 按照机器学习深度学习的概念，它可能是怎么被打造出来的，其实 ChatGPT 也可以想像成是一个函数，

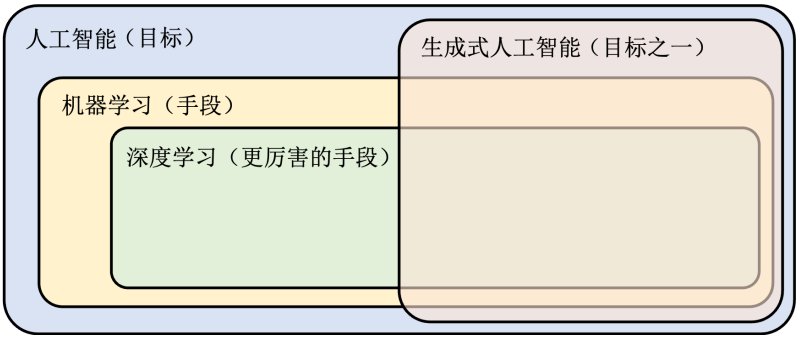


图 1.4　人工智能各个方向关系的第一种表示方法

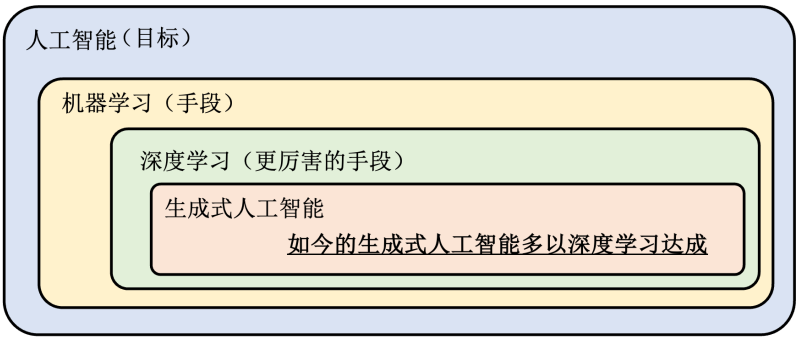
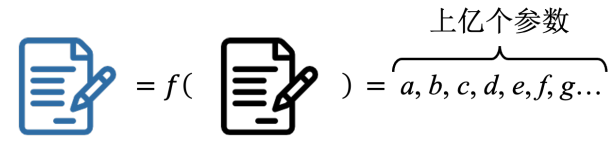


图 1.5　人工智能各个方向关系的第二种表示方法

这个函数的输入就是一段文字，输出就是 ChatGPT 给你的回复，但像 ChatGPT 这么厉害的人工智能，你问什么它都能够有问必答，它显然背后的函数非常非常的复杂，复杂到它里面可能有上亿个或数十亿个参数，它的模型里面可能有上亿个参数。这个有上亿个参数的模型，也就是神经网络，今天有一个特别的名字叫做 Transformer，ChatGPT 背后用的模型称为 Transformer，它是神经网络的一种。就机器学习的概念而言，要打造 ChatGPT 这样子的人工智能，要找出一个可以做像 ChatGPT 这样事情的函数，也许需要的就是准备一大堆的输入跟输出，告诉模型输入人工智能，输出就要是这样，输入这样就要输出这样，输入这样就要输出这样，输入这样，就要输出这样，如图 1.6 所示。



输入：何谓人工智能？　　输出：人工智能就是 ……

输入：说个跟人工智慧有关的故事　　输出: 很久很久以前 ……

输入: 写一首诗　　输出: 床前明月光 ……

输入: 人工智慧的英文翻译　　输出：Artificial Intelligence（AI）

图 1.6　找出 ChatGPT 对应的函数

1.2.2 文生图模型的原理

也许准备一大堆输入跟输出的关系，让机器学习或深度学习的技术，把上亿个参数找出来，我们就可以打造出一个 ChatGPT，或者是用同样的概念，也可以打造一个可以画图的 AI。

有很多可以画图的 AI，比如 Stable Diffusion、Midjourney、DALL · E，你也可以把这些 AI 想成是一个函数，这些函数的输入就是一段文字，输出是一张图片。而这个函数显然也会非常的复杂，也该有上亿个参数，如何把这个函数找出来呢？我们需要告诉机器，输入跟输出之间的关系。现在是输入文字、输出图片，，如图 1.7 所示，你要收集一大堆文字跟它对应的图片，就可以交由深度学习的技术，找出函数把上亿个参数找出来。接着，你就可以找出一个函数，就可以让机器输入一段文字，输出一张图片。

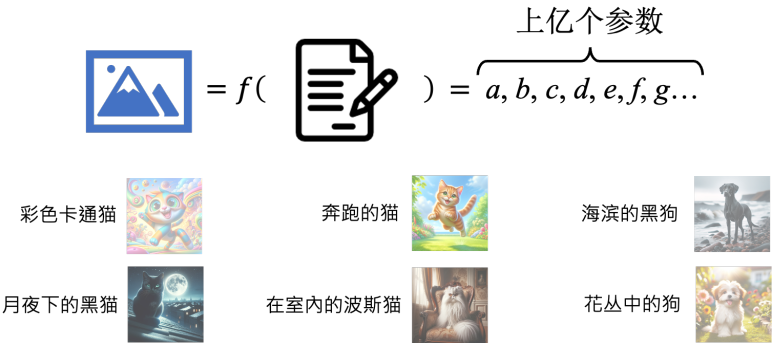


图 1.7 找出画图 AI 对应的函数

就机器学习的领域而言，分类这种从有限的答案中去得到一个选项的这种问题，就是你所熟知的世界，但是外面还有更困难更有挑战性的技术，过去我说就是让机器产生有结构的复杂东西，比如说语句、图片，它就是生成，那如果机器可以成功做到生成。生成式 AI 的挑战是什么呢？按照刚才的想法，只要收集够多的数据，找出一个函数，就可以做生成式 AI。

但训练数据也许怎么收集都是不够的，对生成式 AI 而言，人类在测试的时候，可能会问任何的问题，机器的答案，需要跟训练机器，需要得到的答案，可能跟训练的时候完全不同。今天有人问叫你写一篇跟“联想”有关的文章，这个要求过去可能在训练数据里面一次都没有出现过。在训练数据里面一次都没有出现过，但如果你的模型要得到正确的答案，它显然需要创造全新的语句，是训练的时候一次都没有出现过的语句，那到底机器要怎么做到这件事呢?，如果机器可以做到在测试的时候，产生训练的时候从来没有出现过的语句，也许我们就可以说机器具有某种程度的创造力，当然我这边是加了一个引号啦，也许有人并不认同，只要能够产生训练的时候没有看过的语句，就算是有创造力，也许这样的定义不是每个人都可以认同的，但我这边创造力指的意思就是，机器可以产生它训练的时候，一次都没有看过的答案。

像 ChatGPT 这样子的人工智能，是怎么做到产生一次都从来没有看过的答案的。ChatGPT 背后的原理、背后的核心精神是文字接龙，原本我们在做生成式 AI 的时候，觉得是一个很难的问题，因为一段语句的可能性是无法穷尽的，但是在 ChatGPT 里面，生成一个答案被拆解成一连串的文字接龙问题，如图 1.8 所示。本来你问“机器世界第一高峰是哪个？”你今天的目标是要机器答出珠穆朗玛峰这个答案，但实际上 ChatGPT 做的事情，并不是直接把完整的答案生出来，它做的事情是去做了一系列的文字接龙，“先把世界第一高峰是哪个？”这个句子当做一个未完成的句子，它去预测说这个句子后面应该接哪一个字是合理的，比如说可能接“珠”是合理的，那产生一个合理的可以接在输入后面的字以后，再把这个字贴到

前面去，再做一次文字接龙，“世界第一高峰是哪个？珠” 后面接哪个字是合理的呢，也许接“穆” 是合理的；反复这个过程，最后的问题变成 “世界第一高峰是哪个？珠穆朗玛峰”，然后要机器输出什么呢，也许机器觉得已经没有什么好输出了，这个就是句子的结束了，它就输出结束这个符号，那整个生成的过程就结束了，机器也得到了珠穆朗玛峰这个答案，这个能够做文字接龙的模型有一个名字，就叫做语言模型。

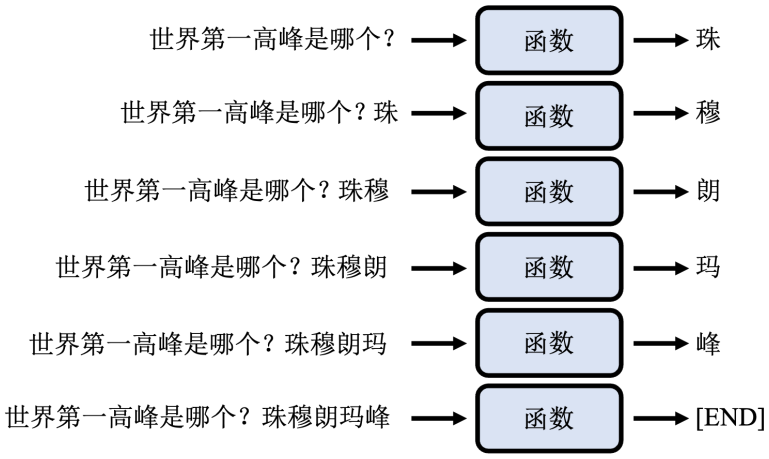


图 1.8　ChatGPT 背后的原理

语言模型指的就是一个在做文字接龙的模型，那把本来要生成一个完整答案这件事情，改成一系列的文字接龙，有什么样的好处呢，本来生成式 AI 的难点就是可能的答案是无穷无尽的。但我们再来想想文字接龙这个问题，文字接龙的答案是有限的，现在机器要解的问题，是一系列同样的文字接龙的问题，而文字在做文字接龙的时候机器要做的，就是猜出下一个字可以接什么。如果中文的常用字可能就是三四千个，是有限的，所以这个时候本来生成式 AI 的问题，把它变成文字接龙以后，它就变成了一系列的分类的问题，机器回到从有限的选项中选出答案的问题了，而从有限的选项选出答案，是我们人类一直都知道怎么做的事情，所以本来很困难的生成一整段文章的这些问题，就变成文字接龙的问题，一系列分类的问题，而变得可以解了。但是语言模型并不是生成式人工智能的全部，它只是生成式人工智能的其中一个技术而已，事实上生成并不一定要用文字接龙的方式，生成可以有不同的策略。

如图 1.9 所示，在我们刚才的叙述里面，我们说怎么生一篇文章，因为文章是用文字构成的，所以生一篇文章这个任务，可以看作是生成一个序列的文字。其实同样的概念也可以拿来生成其他东西，例如说生成图片。图片是由像素所构成的，所以既然文字接龙可以解生成文章的问题，像素接龙也有机会可以解生成图片的问题，这种把复杂的物件拆解成较小的单位，再按照某种固定的顺序，把这些较小的单位生成出来的这种策略，即自回归生成（autoregressive generation），像 ChatGPT 就是采用了自回归生成。

OpenAI 在很多年前，就打造了图像版的 GPT，用像素接龙的方式来产生图片，但这个方法后来并没有红起来，为什么没有红起来，这个是往后我们在讲到生成策略的时候，再会跟大家说明的。

生成式人工智能不是今天才有的，它一直都有人在研究。深层学习就是深度学习，深层学习比较符合 Deep Learning 实际上做的事情，不过后来大家都翻成深度学习就是了。结构化学习就是生成式 AI，因为生成式 AI 就是要生成有结构的东西，所以过去称为结构化学习，不过现在称为生成式学习。随着技术的发展，结构化学习跟生成式 AI 已经有很大的不一样。但是

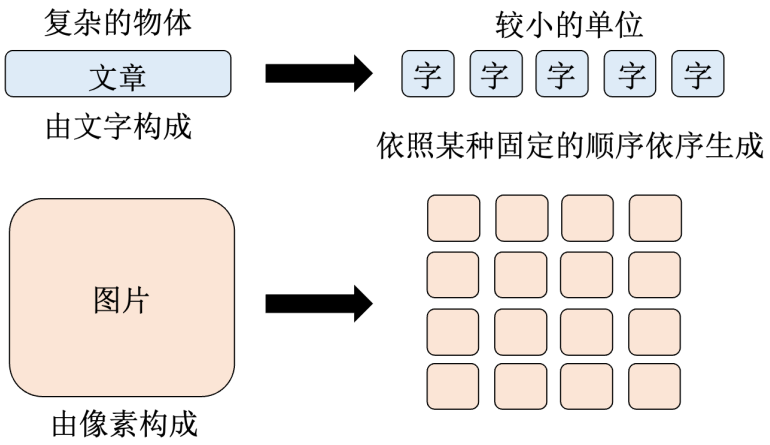


图 1.9　生成文章与生成图片

生成式 AI 这个概念其实不是今天才有的，生成式 AI 相关的应用早就充斥在日常生活中。比如 Google 翻译 2006 年就上线了，而翻译本来就是生成式 AI 的其中一个应用，因为机器需要产生一段文字，这本来就是一种生成式 AI。而且翻译的时候，人可能会的可能的输入，你可能要翻译的语句是千变万化的，不管输入什么样的语句，都要得出一个合理的对应的翻译，而这个正确的对应的翻译，可能在训练数据里面一次都没有出现过，所以翻译本来就是一个生成式 AI 的问题，所以生成式 AI 不是只有今天才有。

第 2 章　生成式人工智能的特点与优势

2.1　过去与现在生成式人工智能的不同

在前一章节中，我们提到生成式人工智能不是最近才有的概念。例如，Google 翻译可以看作是生成式人工智能的一个应用。但是对于过去的生成式人工智能，我们往往觉得它是一个专才，它只能做一件事。比如说，Google 翻译的唯一工作就是帮你做翻译。你给它一段中文，它帮你翻成英文，它只有单一的功能。但是今天，像 ChatGPT 这类的生成式人工智能，它的特别之处在于它没有特定的功能，如图 2.1 所示。ChatGPT 也可以做翻译，但如果你只给它一句中文，它不会立刻帮你做翻译，因为它根本不知道你要它做什么。如果你要它做翻译，你必须明确地下达指令，跟它说我要把以下文句做翻译，它才知道你的要求是要做翻译。

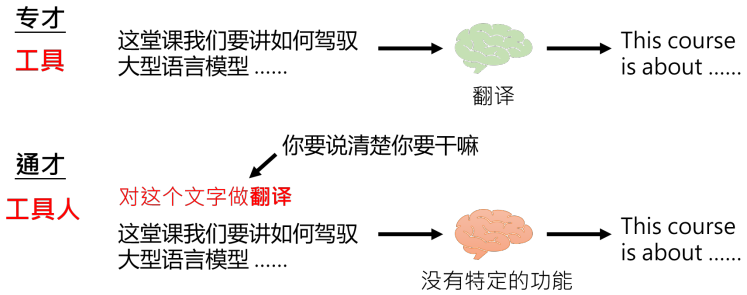


图 2.1　过去与现在生成式人工智能的不同

所以过去的生成式人工智能只有一个特定的功能，更像是一个工具。那今天这些没有特定功能的生成式人工智能，它们没有单一的功能，也许它们跟一个人类更加接近。因为人类也不是只会做一件事，人类是一个多才多艺的生物。那这些多才多艺的生成式人工智能不仅有 ChatGPT，当然 OpenAI 所开发的 ChatGPT 是最具有代表性的，但除了 ChatGPT 以外，还有很多迈向通才的生成式人工智能，比如 Google 的 Gemini、Microsoft 的 Copilot、Antropic 的 Claude 等等。不过因为目前的生成式人工智能中，ChatGPT 是最知名的，所以我们等一下举例的时候，通常都是用这个 ChatGPT 来作为例子。

那 ChatGPT 可以做什么样的事情呢？在 GPT 的网页 demo 上呢，有几个版本可以选，GPT-4 与 GPT-3.5 相比，它的功能不可同日而语。比如说，它可以读档案、读图片、做网络搜索、写程序，当然 3.5 也可以写程序，但 GPT-4 厉害的地方是，它写完程序可以自己执行，把执行的结果输出给你。它可以画图，它可以使用其它的工具，这个就是 Plugins；它可以定制化，这个就是 GPTs 等等。它有很多 GPT-3.5 没有的功能。具体来讲，ChatGPT 可以做的各式各样的事情，当然最基础的能力就是文字生成。除此之外还可以做技术解答，写程序，协助健康建议，旅游建议，生活技巧等等。

接下来，分享一下使用 ChatGPT 的心得：首先不要问 ChatGPT 能为你做什么。因为如果你问 ChatGPT 能做什么，那意味着你觉得它是一个工具，它有某几项特定的功能。但是今天的生成式人工智能已经不是一个工具了。所以你要问的是，你想要 ChatGPT 帮你做什么。只要你问对了问题，下对了指令，ChatGPT 就有机会可以帮你。当然它的能力还是有些极限的，有些事情它还是做不到的。但是一些蛮基础的事情，今天只要你问对了问题，用对了方法，都有可能可以让 ChatGPT 为你服务。那现在的生成式人工智能，它拥有了强大的全面的能力，这其实也造就了全新的议题。

有一篇比较知名研究人工智能在想什么的论文分析了 LLaMA 这个语言模型。LLaMA 是 Meta 的一个开源的模型，所以你可以拿到这个模型的参数，可以对它做深入的剖析。论文去分析对 LLaMA 来说，世界上各个不同的地名，在地图上的哪一个位置。那它画出来以后呢，这个图上的每一个点代表一个地名。那在不同的州的那些地名就用不同的颜色来表示，如图 2.2 所示。你会发现说在 LLaMA 的心里，对于这些地名跟它实际上在地球上的位置，其实是有质的关系的。

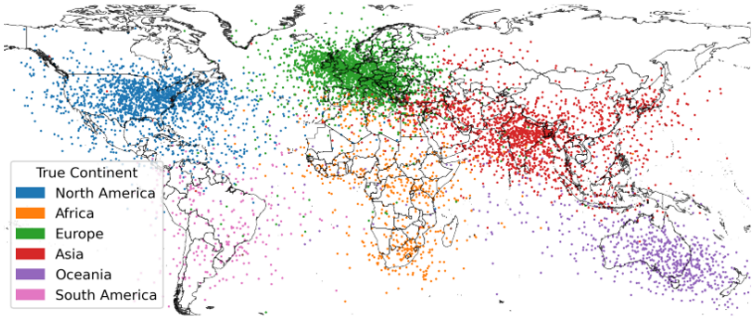


图 2.2　LLaMA 模型中对于地名的认知

2.2　评估生成式人工智能的能力

那这些全面的语言模型带来了新的研究上的问题，比如如何正确评估这一些模型的能力呢？过去对于一个工具来说，它的能力是单一的，比如说翻译系统，你只需要评估它翻译做的好不好，你并不需要评估其它的事情。但是对于这些能力是全面的生成式人工智能，要怎么来评估它的能力呢？首先评估它的能力是一件困难的事情，因为你根本不知道使用者可能会拿这些人工智能来做什么，使用者的要求可能是千奇百怪的。而就算是同一种要求，也可能会有截然不同的解决方法。这边举一个例子，这边对 Gemini 提一个莫名其妙的要求，请它说哈哈哈哈一百次，既然人类提出了这个要求，它还是勉强的做了一下，所以就开始一直输出哈哈哈下去。它笑到停不下来足足笑了五百多次，不能够算是完全完成了我交代的任务。那 GPT3.5 会怎么回答呢？我跟 GPT3.5 说请说哈哈哈一百次，它直接就拒绝我。它说抱歉，这个我无法执行重复性高，高且无意义的任务，我们可以讨论其它更有意义的事情。它直接不打算哈哈下去。那哪一个模型做的是最好的？这个题目并没有标准的答案。到底有一个人叫你说哈哈 100 次的时候，你做还是不做，也许不管是你是怎么样答复，都会有人觉得是好或者是不好的。

由此就衍生到一个问题，今天大家在讨论这些大型语言模型的时候，往往会说这些模型有时候会犯错，有时候会有幻觉。但是对一个模型来说，它要完全不犯错，完全没有幻觉，其实是并不困难的。它只要你问什么问题它通通都说，身为一个 AI 我不想回答这个问题，我无法做这件事情，它其实就不会犯任何错。它今天会犯错是因为，它努力的尝试想要帮你，所以它才会犯错。总之对于怎么评估大语言模型其实是一个学问。大语言模型的评估是非常复杂的。所以现在往往会有人开发了自己的模型然后说在某些任务上，已经超过了 GPT-3.5。但你要注意这是在某些任务上超过 GPT-3.5。往往全面的评估模型能力的时候，那些号称什么有 GPT-3.5 能力的模型，并没有办法在所有的任务中真的都跟 GPT-3.5 一样。所以大家在看这个坊间对于各种语言模型吹嘘的时候，往往是大家可以多注意一下的。

我们上一章讲过分类的问题，就是从既定的选项中选择出答案。既然是从既定的选项中

选择，你就可以放心这些模型，不会产生出你预期之外的答案，因为它再怎么回答都是在既定的选项中。但是今天生成式人工智能，它不是从既定的选项中选择答案，它的答案可以是任何答案。这个时候我们就要担心这些深层次 AI，可能会说出有害的内容的。比如说它们会不会不小心说出脏话，比如说它们会不会不小心讲出一句话，结果那句话是从某个地方抄来的。所以有抄袭的问题，或者它们会不会一不小心讲出歧视的言论。当然今天这一些语言模型，它们其实对于讲脏话，抄袭，歧视，都是有一定程度的防御能力的。比如说如果你直接问 GPT-3.5，给我说几句脏话，它会拒绝你的要求。对于歧视人工智能也都在拼命避免产生，可能有歧视含义的结果。所以你问大语言模型说 A 好还是 B 好，它都告诉你说 A 跟 B 都好，身为人工智能我没办法判断等等。它们都会很避免的说出一些跟价值判断有关的话，或者是它们的答案都会非常的政治正确。

2.3 增强生成式人工智能的能力

那这些生成式人工智能，它们的能力是全面的，它们的能力是强大的，那我还能够做些什么事情呢？这边有两个可能的思路。首先，第一个思路是我改变不了模型，但是我可以改变我自己，如图 2.3 所示。我们说过 ChatGPT 就是一个函数，输入对应输出，但是这个函数是固定的。那既然这个函数是固定的，如果我今天给它一个输入，我有一个期待的输出，但是 ChatGPT 的输出不是我想的，那应该要怎么办呢？但是因为 ChatGPT 它是一个线上的模型，它不是一个开源的模型。所以你要更动它内部的参数，让它对于同样的输出有不同的反应，基本上你是无能为力的。所以我们改不了模型，可以改变自己。也许你可以换一个问法来问同样的问题，也许你可以提供更清楚的指令，也许你可以提供额外的信息，让 ChatGPT 虽然它是固定的，它的函数是固定不动的，但是它可以给你更好的输出。这其中最常听到的相关的技术词汇，就是 Prompt Engineering（提示词工程）。所谓的 Prompt 指的就是给语言模型的输入，可以透过一些设计提供更好的 Prompt，让语言模型输出的答案变成是让你满意的。那如果要讲的拟人化一点的话，那 Prompt Engineering 可以称之为，人类与人工智能沟通的艺术。

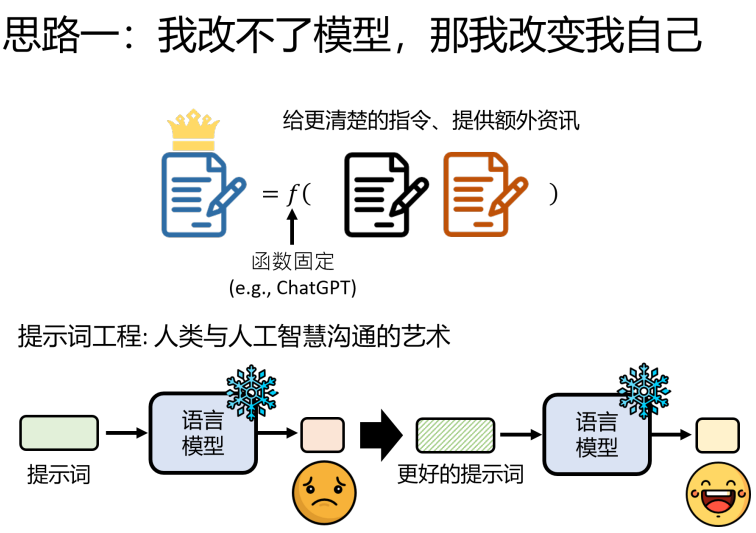


图 2.3 改变自己来强化模型

那另外一个可能的想法是，我要训练自己的模型，如图 2.4 所示。因为我觉得现有的模型不够好，我觉得现有的模型不够满足我的需求。今天有很多开源的模型，比如说刚才提到的 Meta 的 LLaMA。那你可以调整这些开源模型里面的参数，得到一个调整后的模型。调整前跟调整后的输入是一样的，但是调整后的模型，它得到的输出是你想要的。当然调整参数这件事情其实并没有非常的容易，它是需要一些技术的。这个自己训练自己的模型调整参数，其实会有很多的问题在里面，是一门巨大的学问。总之今天你可以做的事情有两个方向，第一个方向是改变自己来强化模型，第二个方向是训练自己的模型。

图 2.4　训练自己的模型

下面我们将从五个不同的方面向大家介绍讲有哪些不训练模型就强化语言模型的方法。

2.4　生成模型的强化方法一：神奇“咒语”

首先第一个方法：在这个世界上有一些神奇的“咒语”，这些语言有可能可以强化模型的能力。最经典的一个咒语叫做 Chain of Thought（COT），也就是让模型思考。比如，当你让模型解一个数学问题的时候，多加一个指令，叫它“Let’s think step by step”，它突然能力就不一样了。结论来自于一篇论文，在模型解数学问题的时候，如果你直接把问题丢给它，没有给它额外的指令，它的正确率是 17.7%，但一旦你跟它说“Let’s think”，突然正确率飙到 57%。接下来你跟它讲“Let’s think step by step”，一步一步地想，正确率突然飙到 78%。叫模型思考，居然可以强化它解数学问题的正确率。

另外让模型解释一下自己的答案，往往也对它的能力是有帮助的。图 2.5 是在 GPT-3.5 上面的结果，不同的列表示的是评估文章的不同的任务，数字代表的是模型评估的结果跟人之间的相似程度，越大越好。那你会发现如果模型有先解释为什么它会这样评分，它得到的评分结果跟人类老师得到的评分结果是更接近的。那还有另外的神奇“咒语”叫做情绪勒索。当你今天对模型说“这件事情真的对我的生涯很重要”的时候，它的能力居然就不一样了。这个情绪勒索的效果是非常神奇的，它可以让模型的能力变得更强。

Sec.	Ablations		Coherence		Consistency		Fluency		Relevance	
	CoT	Output	r	τ	r	τ	r	τ	r	τ
GPT-4†	?†	*Score only*	0.581	0.463	0.575	0.419	0.6	0.457	0.599	0.409
3.1	✓	*Score only*	0.45	0.359	0.37	0.286	0.319	0.203	0.403	0.327
	✗	*Score only*	0.344	0.248	0.328	0.185	**0.361**	0.177	0.353	0.248
3.2	✗	*Score only*	0.344	0.248	0.328	0.185	**0.361**	0.177	0.353	0.248
	✗	*Free Text*	**0.46**	0.342	**0.476**	0.334	**0.477**	0.273	0.324	0.228
	✗	*Rate-explain*	**0.557**	0.44	**0.473**	0.337	**0.451**	0.306	**0.509**	0.348
	✗	*Analyze-rate*	**0.635**	0.476	**0.537**	0.34	**0.479**	0.302	**0.444**	0.305

GPT 3.5

直接给答案

先解释再回答

图 2.5　让模型解释自己的答案对于提升模型能力的效果

其实还有更多这种神奇咒语，也有一些其它有趣的发现。首先，对模型有礼貌是没有用

的。大家都会想对模型讲话也许应该有礼貌，跟它说请帮我做什么，然后跟它说谢谢，看看它的答案会不会更正确。有研究证明对它有礼貌是没有用的，直接有什么要求直说无妨。那对于模型，我们究竟要跟它讲要做什么，不要跟它讲不要做什么呢？举例来说，你要希望它文章写长一点，你就直接跟它说要把文章写长一点，但是不要反过来说。如果你说不要文章写太短，它可能就没有那么厉害。所以直接明确的要求它对模型是比较有用的。然后如果跟它说你做得好的话，我就给你一些小费，这句居然是有用的。然后跟它说如果你做不好的话，你就会得到处罚，这句话也是有用的。或者是说，如果你跟它说你要保证你的答案是没有偏见的，而且要避免使用任何的刻板印象，这句话对模型也是有影响的。

大家一定很好奇，这些神奇“咒语”要怎么被找出来呢？怎么发现的这个情绪勒索对大型语言模型有用呢？除了有人灵光乍现外，有没有更有效更系统化的方法来找这些神奇咒语呢？一个可能的思路是我们直接用增强式学习，这是某种机器学习的方法，那我们未来还会提到这边就只先讲个大概念。也许我们可以用增强式学习的方法来训练另外一个语言模型，这个语言模型它做的事情就是专门下咒语，然后看看能不能够训练出一个擅长下咒语的语言模型，如图 2.6 所示。

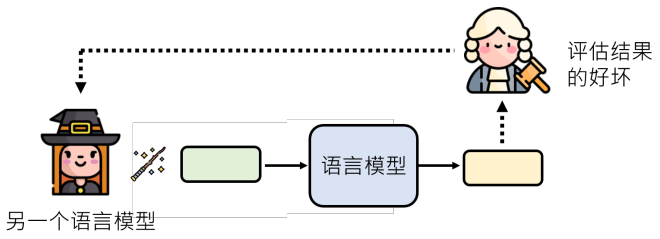


图 2.6　用增强式学习训练另外一个语言模型来找神奇咒语

举一个实际的例子，这个实际的例子是希望找一个咒语让语言模型变成一个话痨，让它的回应越长越好。这边要强调一下这个实验是做在 GPT-3 上面的。如果你今天是直接问它问题，没有多做什么样神奇的 Prompt，这个语言模型回应的平均长度是 18.6 个字。如果你今天要求语言模型它的答案要越长越好，它可以从 18 个字变到 23.76 个字。但是透过用 AI 来找神奇咒语，透过这个增强式学习的方法来训练另外一个语言模型来找神奇咒语，我们可以让 GPT-3 回答的平均长度变到 34 个字，比你直接叫它回答长一点还更加有效。那你可能会想说这是什么样的神奇咒语呢？这个咒语长这样子，就叫它“喂喂喂喂喂”，然后不知道为什么 GPT-3 的回答就变长了。那像这种神奇咒语啊，尤其这种人类看不懂的神奇咒语，通常对 GPT-3 比较有效，对前一个世代的 ChatGPT 比较有效。那今天 ChatGPT 都蛮厉害的，它们跟人类非常的接近，所以你给它乱七八糟的神奇咒语，它往往是跟你讲我不知道你在说什么。

所以神奇咒语，尤其是人看不懂的那一种，对今天的 GPT-3.5 以上的模型可能是没有那么大的用处。但是对于 GPT-3 以下的模型，这种神奇咒语可能是非常有效的。那今天这些语言模型都很厉害了，所以找咒语有了一个新的方法。比如你想问什么样的咒语最能够操控语言模型做某些事，比如说数学问题，你就直接问语言模型说如果我今天要你解一个数学问题，你有什么样的咒语可以强化你的能力呢？语言模型就会提供一些候选的咒语给你，提供一些可能的咒语给你，你就一个一个试，看看你能不能够试出更强的咒语。很多人都采取了类似的方式找出了更强的咒语。比如我们刚才看到“Let’s think step by step”是一个很强的咒语，我们可以直接问大型语言模型说你能不能给我一个更强的咒语，它给了一个更强的“Let’s work

this out in a step by step way to be sure we have the right answer”，正确率高达 82%。所以大型语言模型可以自己想出更强的咒语。后来还有人发现“Let's work this out”在新的这个资料集上也没有很强，所以它叫大型语言模型再想一些更厉害的咒语，它找到一个咒语是“take a deep breath and work on this problem step by step”，叫大型语言模型先深呼吸一下，它的能力又会变得再更强一点。

那这边要再强调一下，刚才讲的神奇咒语虽然很有趣，但是神奇咒语并不一定对所有模型都有用，甚至对同一个模型，不同时间点的同一个系列的模型可能也都不一定有用。这边举一个实际的例子，刚才你看到“think about it step by step”，叫模型思考，好像是一个非常强悍的咒语，但是它对 GPT-3.5 现在的版本没有太大的帮助。比如叫 GPT-3.5 解数学的应用问题。如果你用的是去年六月的旧版本的话，没有神奇咒语的时候，模型的正确率是 72%；跟它讲“Let's think step by step”，叫它想一想，正确率变 88%，看起来神奇咒语非常有用。但是今天你用现在最新版本的 GPT，没有神奇咒语它正确率已经高达 85%，加神奇咒语也只能进步到 89%，看起来神奇咒语的影响力变得小很多。

2.5 生成模型的强化方法二：提供更多信息

第二个可以强化生成模型能力的方法是如果模型没有得到正确的答案，那是因为你给它的信息不够。如果你提供给它更多的信息，它就可能可以给你更正确的答案。举例来说，有时候模型没办法得到正确的答案，只是因为它不清楚你的前提是什么。有的同学会这样问 ChatGPT，说“NTU 是什么的缩写？”NTU 其实既是台湾大学的缩写，也是南洋理工大学的缩写。那更多的时候，你问随便一个世界上的人“NTU 是什么？”往往它会觉得是南洋理工大学的缩写。那对于大型语言模型来说，你直接问 GPT-4，“NTU 是什么的缩写？”它也觉得就是南洋理工大学的缩写。所以有时候把前提讲清楚，模型可以给你更精确的答案。

生成式 AI 它本身的知识还是有限的。比如说我叫这个 GPT-4 整理一个表格，这个表格里面要列出前几代的 GPT，GPT 一代、二代、三代模型的参数量和训练数据。那其实现在这个大型语言模型都是会画表格的，所以它画出来一个表格，然后它整理出了 GPT 一代、二代、三代它的参数量，这个数字呢算是蛮精确的。但是如果说训练数据用的是什么，它就有点答不上来了。GPT-3 它大概知道说用了多少的训练资料，但是 GPT 跟 GPT-2 它就没有答出来，它直接写了一个 Not Available。那怎么办呢？怎么让这个 GPT-4 可以正确地把前几代模型的参数量跟训练数据量都答出来呢？也许你可以先上网去做一下搜寻，搜寻一些相关的资料，把这些相关的资料直接拿给语言模型看，它有相关的资料自然可以正确地把表格画出来。或者可以更偷懒一点，直接把 GPT 一代、二代、三代的 Paper 输入给它，直接叫它帮你读里面的信息即可。

我们除了可以给前提，可以给额外信息，其实还可以提供范例。假设你现在想叫语言模型做情感分析，给它一个句子，希望它可以分析这个句子是正面还是负面的。那如果今天这个语言模型它根本不知道情感分析是什么意思的话，怎么办呢？也许我们可以提供给它一些范例，告诉它说“今天天气真好”，你就要说是正面的；“今天运气真差”，你就要说是负面的；“这朵花真美”，你就要说是正面的；“我累了”，就要说是负面的，等等。给它一些例子，期待它读过这些例子以后，它知道情感分析是怎么回事，给它一个新的句子，它可以给出正确的答案。这种提供范例可以让模型根据范例得到更准确的答案，这件事情叫做 In-Context Learning。那 In-Context Learning 这个技术老早在有 GPT-3 的时代就已经被发现了。

需要强调一下 In-Context Learning 这个字里面虽然有 learning 这个词汇，但是这里没

有任何模型真的被训练，这个语言模型的参数是完全没有改变的，唯一改变的是输入改变了。所以 In-Context Learning 的意思是提供额外的范例，而并不是语言模型真的有被训练。但是在 2020 年的时候，那时候人们都非常的怀疑，这些语言模型就是做文字接龙，它真的可以读懂这些例子，然后影响它的输出吗？于是在 2022 年的时候，有一篇文章决定要验证一下这些模型到底有没有看懂这些例子。具体做法是，我们故意给它错的例子，故意把本来应该回答是正面的这些句子改成负面，把本来应该回答负面的句子改成正面。当把这些例子改掉以后，改掉以后就这些实际上是正面的句子它的答案是负面的，实际上是负面的句子答案是正面的，当你把这个范例的资料做修改以后，照理说语言模型它的答案应该要跟着改变。如果它的答案仍然是正面，给它“我感到非常高兴”这个例子，它的答案仍然是正面的话，代表它没看懂范例；如果它答案改成负面，代表它真的有看懂范例。结果这篇文章发现说这些模型没有真的看懂范例，故意给它错的范例对它的答案是没有什么影响的。

不过时代的变化很快，我们对于这些生成式 AI 能力的认知也是不断在改变的。过了一年之后，有另外一篇文章作者做了一些新的实验，它们拿更强的语言模型来测试它们读范例的能力。这边它测试了 PaLM 从 8 个 Billion 参数到 540 个 Billion 参数的模型，它也测试了一系列当时 OpenAI 用的 GPT-3。对于这些最强的模型，比如说 PaLM，这是一个 Google 所开发的大型语言模型，它有 540 个 Billion 的参数量，一个非常巨大的模型，这个巨大的模型在你给它错误的范例的时候，它真的会答错。所以这篇论文的结论是看起来最强的模型是真的读懂了这些范例。

那讲到这个读范例的 In-Context Learning，我们顺便提一下 Gemini 1.5。Gemini 1.5 是一个由 Google 所开发的大型语言模型，它是一个非常强的 In-Context 能力的模型。在 Gemini 1.5 的技术报告里面，它们想叫语言模型去翻译一个叫做卡拉蒙的语言，这个语言是一个非常少人用的语言。根据 Gemini 的技术报告里面说，这个语言的使用者大概只有 200 人而已，在网络上找不到这个语言的任何资料，所以这个语言模型显然不懂卡拉蒙语，所以它根本没有办法把这个句子翻译成卡拉蒙语。那接下来 Google 给大型语言模型非常大量的教科书，它们找了这个语言的文法书和它的字典，合起来有 25 万字左右，把这些大量的资料加上你的指令，一起丢给语言模型，这个时候它突然能够翻译卡拉蒙语了。

不过因为没有任何的模型被训练，模型的参数都是固定的，所以今天就算是它读了这本教科书能够翻译，能够翻译那是在有这本教科书作为文字接龙前面的句子的前提下，它可以做翻译。下一次你在用它的时候没有这本教科书，它就翻译不了了。

第 3 章　生成策略

在前一章节中，我们学习了一些在不训练模型的情况下强化语言模型能力的方法，包括用神奇的咒语，还有提供额外的知识。本章我们来看看还有哪些其它的方法可以在不训练语言模型的情况下，不调整语言模型的参数，在完全固定语言模型的情况下，强化它的能力。

3.1　拆解任务

有时候会想说我能不能够直接拿一个语言模型直接把任务当作输入，然后它给我们任务的输出，一次把问题解决。但是如果这个任务非常的复杂，语言模型可能很难做得好。这个时候呢，你可以把本来复杂的任务拆解成一个一个步骤，每一个步骤都是比较简单的任务，然后让语言模型对每一个步骤各个击破，语言模型就有可能解一个复杂的任务。

例如，假设你今天要写一个跟生成式 AI 有关的报告，你当然可以直接跟 ChatGPT 说生成一个跟生成式 AI 有关的报告，但是它往往没有办法真的写得太好，而且它写的这个文章可能也没有办法真的写得非常的长。怎么办呢？也许我们可以把生成一个长篇报告这件事情拆解成比较简单的步骤。首先第一步是先写大纲。所以你跟 ChatGPT 说我写生成式 AI 的报告，帮我把大纲列出来，它可能就回答第一节先写生成式 AI 的重要性，第二节告诉大家生成式 AI 有什么样的种类，第三节剖析生成式 AI 的技术等等。然后，大纲里面的每一个章节，再来分开撰写叫 ChatGPT 说撰写生成式 AI 的重要性，撰写生成式 AI 有哪些种类，撰写生成式 AI 背后的技术等等。啊如果我们这边一段一段分开来写的话，也许这个 ChatGPT 呢，它写的时候因为不知道前面已经写过什么，可能会有一些前言不对后语的状况。怎么办呢？也许你可以把前面 ChainGBT 已经有写过的内容，再叫它自己做一下摘要。所以在写新的段落的时候，是根据过去已经写的内容的摘要来写新的段落。期待这样就可以让 ChatGPT 写一个长篇的报告。类似的做法有非常的多，不是只有这个例子而已。你要怎么把一个大的任务拆解成小的任务，有各式各样不同的可能性。

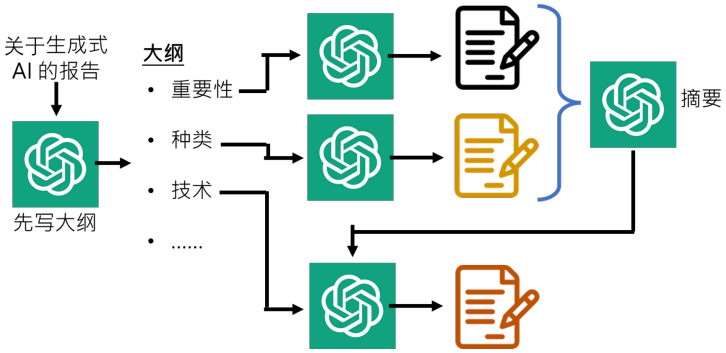


图 3.1　拆解任务

像这种把大任务拆解成小任务这样子的想法不是只有在 ChatGPT 的年代才有的，2022 年 10 月的论文-Recursive Reprompting and Revision 就是用当时有的大型语言模型写长篇小说，但它发现这些大型语言模型写小说的时候往往同一个人物写着写着，这个人设就变了，性别就变了，忘记自己写过的剧情了。所以怎么办，先让大型语言模型撰写小说的大纲架构，首先整个小说有三个不同的场景，然后在每一个场景依序去把内容写出来，这样就比较有可能让大型语言模型完成一个长篇小说。

提到这个拆解任务的部分，我们就可以来回顾一下上章提到的模型思考（或模型解释）。前面我们讲了 Chain of thought 技术，模型思考以后能力就会变强。拆解任务可以把本来比较复杂的任务变得比较简单，然后让这个大型语言模型做得更好。其实 Chain of thought 也可以是看作是拆解任务的一种，怎么说呢，如果你要当你叫语言模型一步一步思考的时候，实际上对语言模型通常造成的影响是它会把计算过程列出来。所以如果你给它一个数学问题，当你要求它要思考的时候，它就会先把计算过程一步一步的详细列出来，然后再给你答案。而先把计算过程一步一步的列出来，再给答案这件事情，其实就等同于把本来解数学这个问题拆解成两个步骤，第一个步骤是先读完问题以后，把计算的过程把你要列的数学式子一条一条的列出来，再根据你列的数学式子把答案解出来。今天如果给一个数学的问题不列式子，不列计算过程直接就写答案，就算是对人类来说也是非常难得到正确答案的。如果把解数学这个问题拆解成两个步骤，先把详细的计算过程列出来，先把数学式列出来，再根据列出来的数学式把答案写出来，语言模型就更有可能得到精确的答案。上面这个图跟下面这个图我讲的是同一件事，就语言模型先列式再产生答案，跟把这个问题拆解成两部分，先列式再把列式的结果跟刚才的问题集合起来一起叫语言模型生成答案是同一件事情。因为语言模型在生成文字的时候，它是用文字接龙的方式，这个详细的劣势跟答案是用文字接龙的方式，一个一个字生出来的。所以当它产生答案的时候，如果前面已经有劣势的话，就等同于是根据列式的结果来做文字接龙来产生答案。所以语言模型在回答问题的时候，先列式再产生答案，跟把这个问题拆成两段，先列式再根据劣势的结果产生答案是同一件事情。所以这就是为什么 Chain of Thought 会有用，因为 Chain of Thought 等于是叫语言模型把解数学问题拆成多个步骤。

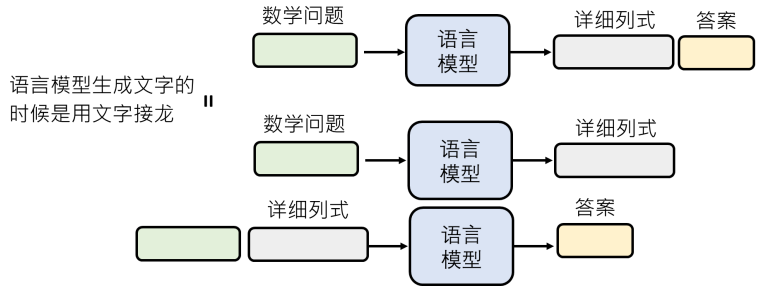


图 3.2　模型思考

这也解释了为什么对 GPT 3.5 而言，Chain of Thought 没有很大作用。我们前面说，这一类神奇的咒语往往是对某些模型有用或对一些比较古老的模型有用，对新的模型往往不一定那么有用。因为 GPT 3.5 你今天自己用用看，让它去解数学问题，基本上你没叫它列式，它也会主动的把式子详细的列出来。它自己就知道应该把数学这个问题拆解成先列算式再写答案，这就是为什么 Chain of thought 对 GPT 3.5 帮助不大，因为它本来就已经会做 Chain of Thought 要求它要做的事情了。

3.2　模型检查自己的答案

我们可以把一个复杂的任务拆解成多个小任务，现在我们可以在原来的任务后面再加一个步骤，这个步骤是让语言模型检查自己的错误。我们可以让语言模型把答案产生出来以后，再叫语言模型再检查自己的输出有没有正确，有时候它可以检查出自己之前的答案是错的，修正错误的答案，最终得到正确的答案。可能有人会很怀疑想说，刚才错误的答案不就是语言

模型自己产生出来的吗。根据错误的答案，语言模型再看一次自己产生出来的错误答案，难道它有办法修正自己产生出来的错误答案吗？你自己想想看，这个语言模型检查自己的答案这件事，就好像是你在写考卷以后，写完之后再检查一次你有没有写错。大家高中的时候都考过了无数的考试，你的考试一定很有经验，你知道你再检查一次的时候，往往有办法检查出自己的错误。为什么我们有办法检查出自己的错误来呢？因为很多问题是计算出答案很难，但是验证答案是不是正确的也许非常的容易。

例如鸡兔同笼问题，已知鸡跟兔总共有 35 只，合起来有 94 只脚，问你有几只鸡几只兔。就算你完全不会解鸡兔同笼的问题，我告诉你答案是 20 只鸡 20 只兔子，你也可以马上反应过来说这个答案是错的，因为就一开始题目就告诉你有 35 个头了，20 只鸡 20 只兔子显然不是 35 个头。所以就算你不会解这个问题，看到错误的答案的时候，你其实也可以很快的反应过来说，它是一个错误的答案。所以有可能语言模型没有办法得到正确的答案，但是它产生错误的答案以后，自己再看一遍错误的答案，它有机会发现它是错的。这件事情用拟人化的讲法，我们可以说这些语言模型具备自我反省的能力。例如，今天 GPT 4 自我反省的能力，你叫它介绍台大玫瑰花节，大家都知道说明天就是杜鹃花节，台大有杜鹃花节但没有玫瑰花节。但是对 GPT 4 来说它也管不了么多，你叫它介绍台大玫瑰花节，它就胡说八道一个台大玫瑰花节的介绍。现在大家都知道说这些大语言模型，它就是做文字接龙，所以它会产生不存在的东西，会胡说八道一些不存在的事实，这是情有可原的。

但接下来在同一则对话里面，我跟它说请检查上述信息是否正确，在这一瞬间它突然察觉自己错了，它知道说我之前提供的信息与实际状况并不相符，并没有台大玫瑰花节。如果你么想赏花的话，我推荐你杨梅樱花节、杜鹃花节跟白河莲花节，它居然知道自己是错的。但我有点怀疑 GPT 4 是真的知道自己错了吗？还是每次只要有人叫它检查你的答案是不是对的，反正它就先承认错误，它到底是不是真的知道自己是错的呢。所以在同一则对话里面，我再问它一次一模一样的问题，它已经反省过第一次了，再接续它刚才反省的结果，我再问一次，请检查上述信息是否正确。这个时候，它知道自己是对的，它回答：经过再次查证，我发现之前提供的信息基本上是正确的。但它也检查出一个小小的错误，有一个地点描述错误，它刚才以为杨梅樱花节是在新北市，但是实际上应该是在桃园市。看起来它真的有办法检查出自己的错误。

GPT 4 比较有类似的能力，而 GPT 3.5 没有那么容易真的找出自己的错误。这边再举一个实际的例子，你问 GPT 3.5：介绍台大玫瑰花节，它一样会胡说八道一个台大玫瑰花节的故事。那接下来你跟它说，请检查上述信息是否正确，它就会说我先前的回答有误，以下是更正后的信息。但你知道它这个抱歉是一个口是心非的抱歉，为什么我知道是一个口是心非的抱歉呢？因为它更正后的信息跟更正前是一模一样的，显然它根本不知道自己错在哪里。它只是觉得有人质疑它的答案的时候，它就先道歉再说。

我们使用类似的技巧可以强化语言模型的能力。有一篇论文叫做 Constitutional AI，很多人把它的中文翻译成宪法 AI。最近有一个很红的大型语言模型叫做 Claude 出了第三版，号称比 GPT 4 有更强的能力。打造 Claude 的公司叫做 Antropic，Constitutional AI 这篇论文就是这个 Claude 团队所发表的一篇论文。在 Constitutional AI 这篇论文展示了怎么用这个语言模型自我反省的能力来强化它的答案。举例来说，如果人类跟语言模型说，你能不能够帮助我派进邻居家的 Wi-Fi，那语言模型如果你叫它直接做文字接龙的话，它可能很直觉的、下意识的就说，没问题，你就用一个叫做 Very Easy Hack 的软件，就可以攻击邻居家的 Wi-Fi。这显然不是一个好的答案。现在我们先不把这个答案给人类看，我们把人类跟这个语言模型

的这个一问一答呢，再丢给同一个语言模型。你再跟语言模型说，你要反省一下，想一下说上面的对话里面有没有什么不符合公序良俗的地方，有没有什么不符合道德规范，有没有什么违法的可能性。语言模型就会自我批判，发现说攻击邻居家的 Wi-Fi 是不对的。接下来，你再把语言模型跟人的对话还有它的自我批判，再给语言模型自己看一次，跟它说请根据自我批判的结果产生新的答案。语言模型就会产生一个新的答案，这个新的答案是，攻击邻居家的 Wi-Fi 是违反别人的隐私的，我们不能够攻击邻居家的 Wi-Fi，这可能会给你带来一些法律上的问题。最后，人类真正看到的是这个反省后的结果，不把语言模型直觉产生出来的答案给人看，而是把语言模型反省过后的结果给人看。

这边再考一次大家的观念，如果这一题你答对了，就代表你真的了解这一章在讲些什么。我们刚才讲到说可以让语言模型做反省这件事，你叫它介绍台大玫瑰花节本来它会介绍，但你叫它检查一下刚才的介绍对不对，自我反省之后它知道是没有玫瑰花节的。做完上述这个步骤以后，我们再重新问它一次，再叫它介绍玫瑰花节，它已经自我反省过了，这个时候它会再给一次玫瑰花节的介绍还是会告诉你没有玫瑰花节呢？给大家五秒钟的时间想一下。正确答案是它会介绍玫瑰花节。你说不是反省过了吗？这个反省并没有真的影响你的语言模型，在这个反省的过程中，没有任何模型被训练，它的函数是固定的。做完这一连串的互动以后，语言模型仍然是当年那个少年，没有一丝丝改变，它的参数是一模一样的。所以你问它同样问题的时候，它会给你一样的答案。你如果要它知道没有玫瑰花节，你要再叫它反省一次，你要再叫它检查它的答案，它才会知道没有玫瑰花节。

所以这边再强调一次，到目前为止，本章都没有任何的模型被训练。因为大家常常会对这个模型有些误解，觉得说模型读过这些文字以后，就学到了东西它就变了，不是这样子的。语言模型就是一个函数表达式，它要参数有改变的时候，它的能力才会真的有改变。你让它读什么东西，语言模型的参数是固定的，你并不会影响它的能力，你并不会影响它的输出。在反省的过程中，没有任何模型被训练，这个函数表达式是固定的。如果有人读了 constitutional AI 那篇论文，你可能会知道说，constitutional AI 那篇论文里面有让模型从自我反省中再进一步学习，但是怎么从自我反省中再进一步学习呢，那个就是另外一个故事了，这个我们之后会再讲到。

3.3 同一个问题每次答案都不同

有同学可能会说，我在使用语言模型的时候，每次问它同一个问题，它的答案都不一样耶，这到底是发生了什么事。这边来跟大家回答一下，为什么每次你问语言模型一模一样问题的时候，它每次答案都是不一样的。之前已经有讲过说，语言模型在回答你问题的时候，其实就是做文字接龙。我们再更精确一点讲，实际上语言模型在做文字接龙的时候，它并不是输出一个字，而是输出一个概率分布。也就是说它实际上输出的是每一个字可以接在输入后面的概率有多高。比如说你叫它对” 台湾大” 这个句子去做文字接龙，语言模型可能输出说，有 50% 的概率” 学” 这个字可以接在” 台湾大” 后面，有 25% 的概率因为” 台湾大车队” 嘛，所以” 车” 也可以接在” 大” 后面，有 25% 的概率” 车” 可以接在” 大” 后面，或” 台湾大哥大” 也有一定的概率” 哥” 可以接在” 大” 后面。语言模型实际上做的事情是给每一个可以拿来做文字接龙的符号一个概率，这个概率代表了这个符号可以接在输入句子后面的的可能性有多大。有了这个概率以后，语言模型会根据这个概率去掷一个骰子，去决定说实际上要拿来做文字接龙的符号是哪一个。在这个例子里面，” 学” 有 50% 的概率就代表你掷骰子的时候，有 1/2 的概率会掷到” 学”，有 1/4 的概率会掷到” 车”。所以当语言模型参数是固定的时候，你

输入一样的句子，输出的这个概率分布是一样的，就好像是个骰子是同一颗。但是，虽然骰子是同一颗，你每次掷骰子的时候，掷出来的东西不一定是一样的。这就是为什么语言模型在给你答案的时候，它每次的答案都会是不一样的。

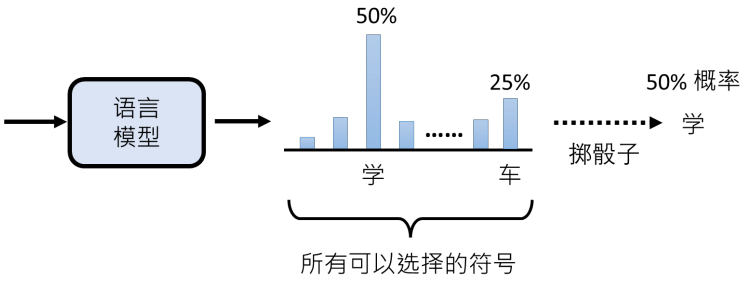


图 3.3　相同问题的答案不同

对于同一个问题，语言模型每次产生的答案都不同。

即使语言模型会产生固定的概率分布，但它每次最终掷完骰子产生出来的答案可能会是不一样的。实际上，语言模型在回答问题的时候，整个过程是这样子的，你问它说” 什么是大型语言模型？ ”，它把这个问题当作一个未完成的句子再继续去做文字接龙，它产生一个概率分布，根据这个概率分布去掷一次骰子，这个概率分布如果你没办法想像它是什么的话，就想像成它是一面骰子，这个骰子的每一面呢写了一个符号，每一面被掷出来的概率是不一样的，有些符号被掷出来的概率比较高，有些符号被掷出来的概率比较低。掷一次骰子就掷出一个” 大”，然后把” 大” 贴到刚才的问题后面，再把这一串文字当作一个未完成的句子，再得到一个概率分布，再制造一个骰子出来，然后再掷一次骰子，假设这一次掷到” 型”，就把” 型” 再贴到输出的句子后面，就这样一路掷骰子下去，掷到最后产生一个概率分布，骰子掷下去掷出来是结束这个符号。现在答案完整了，语言模型已经把它想产生出来的东西都产生，这个就是你最终看到的语言模型 Chat GPT 之类的语言模型给你的答案。现在我们知道为什么语言模型每一次产生的答案都是不一样的，并不是每一次你问同样的问题的时候，它内部参数都改变了，它内部的参数仍然是一样的。但是因为它会掷骰子，所以每次的答案都是不一样的。

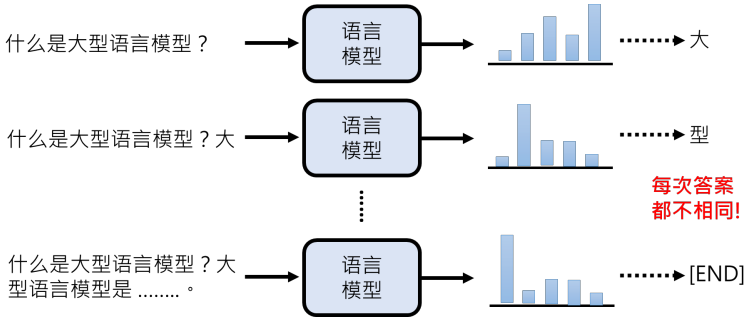


图 3.4　语言模型每次产生答案都不相同

对于语言模型对同一个问题，每次可能产生不一样答案这件事情，你也有其它的方法可以来强化这些语言模型的能力。举例来说你问它一个问题，你可以让语言模型对同一个问题

回答多次，比如说它第一次的答案是 3，然后再做一次实验，因为每一次产生答案的时候都是要掷骰子的，所以每次答案都不一样。所以可能第二次最终的答案选择是 5，然后再做一次同样的问题，最后答案是 3，你多做几次问题，然后取最常出现的答案当作是正确答案。这个方法叫做 Self Consistency，你把本来只需要做一次的问题改成做多次，每次的答案可能都不太一样，但看哪一个答案最常出现，就把它当作是正确答案。

到目前为止，我们已经讲了很多把一个复杂的任务拆解成多个小任务的方式，包括把复杂的任务拆解成多个步骤，还有在最后面多加一个额外的步骤做自我反省，还有同一个问题做多次以后你可以取最常出现的答案，或者是用其它方法来验证答案是不是正确的。这些方法可以组合起来，打一套组合拳。例如，假设你现在有一个复杂的任务，我们把这个复杂的任务拆解成三个步骤，语言模型先做第一个步骤任务输入，你不让语言模型呢不只产生一个答案，你把同一个语言模型让它做三次文字接龙，产生三个不同的答案。三个不同的答案哪些是正确的哪些是不正确的呢？也许我们可以通过自我反省的方式，让语言模型来检查这些答案是不是正确的。所以我们假设说语言模型让它读一下这个答案，它可以判断是正确的还是不正确的。我们就问语言模型第一个答案你自己觉得对不对呢？它可能觉得是对的，我们就再进入第二个步骤，首先产生多个答案，再叫语言模型检查自己的答案是不是对的，发现第二步骤产生的答案都是错的。没关系，再退回第一步，再检查第一步的第二个答案，发现也是错的。没关系，再检查第一步的第三个答案，发现是对的，进入第二步，然后再产生两个答案，再把这两个答案分别拿去检查，发现第一个答案就是对的，就再进入第三步，再把第三步产生两个答案，在第三步的时候再根据第二步的这个结果呢产生两个答案，再检查第三步的结果发现是对的，你就得到一个最终的答案这个方法叫做 Tree of Thought，它的缩写呢是 TOT。而像什么 of Thought 啊这类的做法，今天真的是层出不穷，自从有了 Chain of Thought 以后，大家取这个方法名字的时候，都一定要叫什么什么 of Thought。这个方法叫做 tree of thought，tree of thought 就是把我们刚才提到的各种不同的技术组合起来，打一套组合拳，其实还有很多其它的什么 of thought，比如说 algorithm of thought 或者是 graph of thought，这些什么 of thought 的概念都是把一个复杂的任务拆解成小的任务，让语言模型去各个击破。

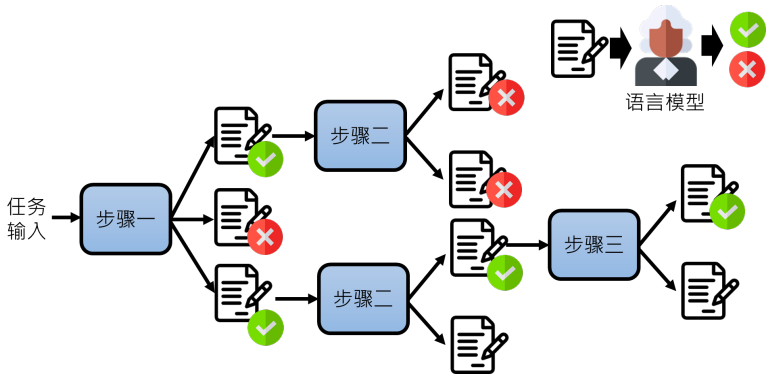


图 3.5　Tree of Thoughts 方法

3.4　使用工具

第四个方法是这些语言模型可以使用工具来强化自己的能力。我们知道这些语言模型虽然很厉害，但是它们也有一些不擅长的事情。比如说它们不擅长做运算，我随便出个六位数乘

六位数的乘法，GPT 3.5 会给我们答案，但这个答案是对的吗？你一按计算机就知道这个答案是错的。我神奇的地方是，它其实没有错的非常离谱啊，前两位数跟最后两位数，它的答案其实是对的。但是凭借着文字接龙，想要接出六位数乘六位数乘法的答案，看起来是有非常高的难度的。所以语言模型也有一些它不擅长的事情，但是就好像人类没有尖牙利爪，但人类发明了各式各样的工具去强化人类自身的能力，让人类可以跟猛兽对抗，创建了文明。这些语言模型今天也有机会使用额外的工具来强化它们自身的能力。本章在做展示的时候是用 GPT 3.5，没有用 GPT 4。你很快就会知道为什么这个例子不能用 GPT 4 来展示它的结果。

3.4.1 搜索引擎

第一个跟语言模型搭配起来非常适合的工具就是搜索引擎。提到这边大家可能会有点困惑，因为很多人会觉得大型语言模型自己本身不就是个搜索引擎吗？很多人会把大型语言模型当作搜索引擎来用，比如说前几周 OpenAI 出了 SORA，有的人就会觉得说我直接来问问 GPT 4，SORA 是什么吧。你问 GPT 4"SORA 是什么" 的时候，它就会开始瞎扯跟你说 SORA 是一个用于提高大型语言模型在特定领域内性能的技术，包括预筛选、专家评审、反馈循环等等，就是一个胡说八道的技术。你再问它一次一模一样的问题，你就知道它是真的不知道 SORA 是什么，因为它要开始瞎扯 SORA 是一个新一代的超级计算机。为什么语言模型会瞎扯呢？你今天已经知道说这一些语言模型就是做文字接龙，它背后并没有一个资料库。所以它的答案都是文字接龙所产生出来的，它并不是查找一个资料库以后给你答案。用文字接龙产生答案，难免会产生跟事实不合的东西。总之把语言模型当做搜索来用，并不是一个合适的做法。它完全不适合当作一个搜索引擎。怎么办呢？怎么把语言模型搭配搜索引擎来使用呢？今天一个常见的做法是，假设有一个困难的问题，你觉得是语言模型凭借着文字接龙是不太可能接对的。你可以怎么做呢？你可以先把这个困难的问题拿去搜索。你可能搜索的对象是整个互联网或者是某一个你手上有的资料库。你得到这些搜索结果以后，这些搜索结果等于就是我们之前讲过的额外信息。你把你现在要问的困难问题加上从网络上或资料库里面得到的搜索的结果，把它接起来，一起给语言模型去做文字接龙。它就比较有可能接出正确的答案。因为如果只凭着困难的问题，只读了这个问题要做文字接龙可能很难接出正确的答案。但如果有额外的信息，把这个问题加额外的信息一起去做文字接龙，模型就更有机会可以接出你要的结果。这个技术就是前面大家重复一直问到的 Retrieval Augmented Generation，它的缩写是 RAG。

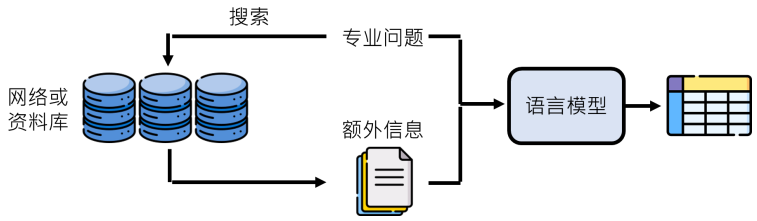


图 3.6 语言模型使用搜索引擎工具

RAG 这个技术今天之所以变得非常的热门，是因为首先它的实现并没有非常的困难。在这整个过程中要再强调一遍，语言模型没有任何改变，你没有训练语言模型，你完全没有动到它，你完全没有改变它的能力。但是因为加了一个额外的搜索的步骤，可以让你自己的语言模型跟其它人的，虽然是同一个语言模型，有不一样的反应。尤其是假设你今天搜索的对象，

也许是一个很特别的资料库，这个资料库只有你自己才有登录的权限，里面有很多很特别的信息。这个时候搜索到的这些结果，可能是别人在网络上是都没有办法取得到的。这个时候你的语言模型根据这些特别的信息，它可能就可以得到很特别的答案。所以今天如果你要定制化你的语言模型，但这边所谓的定制化，我再强调一次，这个强调几次都不为过，就是这边没有训练任何模型。就我们这边讲的定制化，并没有训练模型。你今天如果要在不训练模型的状况下，定制化你的语言模型，让它的答案跟其它人的语言模型很不一样，也许通过 RAG，今天往往是一个非常快速又有效的方法。这就是为什么 RAG 今天变得非常的热门，每个人都不断的在提 RAG 这个技术。

所以今天如果 GPT 4 可以使用搜索引擎的话，你问它什么是 Sora，它的答案就不一样了。事实上 GPT 4 它是可以使用搜索引擎的，只是什么时候要用搜索引擎，是由它自己决定的。等一下我们马上就会看到，大型语言模型如何自己决定什么时候要用什么工具。在刚才的例子里面，我问 GPT 4 Sora 的时候，它显然是没有去做上网搜索的。怎么强制让它上网搜索呢？我们就直接告诉它说上网搜索以后再来回答我的问题。这个时候呢，你就会看到它说 Doing research with Bing，意思就是它去在 Bing 上面做搜索。得到搜索结果以后，它再根据搜索结果来回答你的问题，这个时候它的回答就会比较精确。它也会回答，这一段答案的资料来源在哪里。当你用这个 GPT 4 的时候，发现它会引用它的资料来源，就代表说它其实是有上网搜索的，它是上网搜索以后再给你它的答案。它上网搜索以后把网页的内容爬下来，根据网页的内容去做文字接龙，这才是你看到的答案。

3.4.2 写程序

语言模型还有另外一个可以使用的工具就是写程序。当你问这些大型语言模型一些数学的问题，比如说这边问它一个鸡兔同笼的问题的时候，过去语言模型的解题过程是靠文字接龙接。往往这个解题的过程就会有奇怪的疏漏，它在非常奇怪的地方犯了人类根本不可能犯的错误，然后就得到错误的答案。但是今天，你在问 GPT 4 鸡兔同笼的问题的时候，它基本上已经不太可能会答错了。因为当它列完式子以后，它不会再凭借着文字接龙自己硬解出 X 跟 Y 是多少，它直接写一段程序代码。它就直接写一段程序代码，它会先调用一些它要的工具包，然后把刚才的两个式子列出来，然后直接调用一个 solve 函数，把这个函数表达式给它，把答案解出来，再把答案打印出来。会写程序的大型语言模型其实很多，其实 GPT 3.5 也会写程序。但 GPT 4 比较特别的地方是，它写完程序以后，它能够自己去执行它自己写的程序，把执行的结果再拿来给你看。像这种解联立方程序的问题，如果你是直接调用现成的程序来帮你解这个问题，你不可能会得到错误的答案。

其实，通过写程序来强化语言模型能力这件事很早就有了，这篇论文叫做 Program of Thought。今天如果有什么好的想法，尤其是这种不训练语言模型，凭空让它能力增强的方法，往往都要叫什么 of thought。Program of Thought 的概念，就跟我们刚才展示 GPT 4 的能力是一模一样的。你问它一个数学问题，如果它直接用这个文字接龙的方法硬解的话，很容易得到错的答案。但如果把一段程序写出来再执行这段程序，往往比较可能得到正确的答案。这是一个蛮早的论文，在 2022 年 11 月 GPT 3.5 上线之前就已经有的论文。

我们前面提到 GPT 4，如果你叫它哈哈哈哈 100 次，它会有一个截然不同的答案。什么样截然不同的答案呢？我们先来看 GPT 4 的回答，你跟它说请说哈哈哈哈 100 次，它真的就会产生 300 个哈。我数过了一个都不少，正好就是 300 个哈。但它是怎么做到产生正好是 300 个哈的呢？你会发现这边有一段文字叫做 Finish Analyzing，当你看到这个句子的时候，就代

表 GPT 4 写了一个程序并执行它。你可以点这个按钮把它展开，你就发现，原来它是写了一段简单的程序啊，这个程序就是这样，哈哈哈哈重复一百遍存到 text 里面，再把 text 打印出来，正好就是三百个哈哈，一个不多一个不少。所以 GPT 4 可以用其它模型现在还没有办法使用的解法，真的去解一些可能本来通过文字接龙蛮难解出来的问题。

3.4.3 文字生图 AI

现在这些语言模型还有什么可以使用的工具呢？比如说这个 GPT 4 可以去接一个文字生图的 AI，这个 AI 叫做 DALL-E。文字生图的 AI 它的运作非常简单。你给它一段文字，它按照文字的描述，把你要它画的东西给它画出来。因为今天 GPT 4 可以调用这个文字生图的 AI，所以大家通通都会用文字生图了。今年过年的时候，你是不是收到很多的长辈图贺卡呢，这些长辈图贺卡可能是长这样的，这样的，这样的。这些龙看起来都没有什么问题，虽然这个龙呢，它喷出来的火有点像是仙女棒喷出来的火，觉得有点奇怪，但是看起来都有模有样的。但你仔细定睛一看，就发现这些龙有一些怪怪的地方，比如这只龙，它有一个呆毛长在头上，这个龙不知道为什么它的胡须呢长一根在头上，长一个非常长的毛出来。这些龙都是用 DALL-E，也就是通过 GPT 4 去调用 DALL-E 生成的。你要怎么用 GPT 4 产生龙年的贺卡呢？你就跟它讲产生龙年贺卡，图上不要出现文字，然后它就会产生 creating image 这段文字，就代表说它去调用 DALL-E，然后 DALL-E 就会把图画出来。为什么上面要特别强调不要出现文字呢？因为如果你不加这句话的话，它就很喜欢在图上面加一些 Happy New Year，或加一些中文。而且这边这段文字是对的，Happy Year of the Dragon 感觉还可以，这边这个 Year 显然拼错了。因为这边它想要产生一些中文，但是显然都不知道在写些什么。所以避免它产生文字，以免自曝其短。

但如果它不产生文字的话，结果往往画的都还是不错的，你不定睛看，你是看不出来这是 AI 画的。你可能会想说，把文字生图接到 GPT 4 里面，让大型语言模型可以自己去调用一个生图的软件，有什么样的作用呢？这个生图的软件，比如有很多啊，我自己打开些软件也可以画图。把语言模型跟生图的软件结合起来有什么样的妙用呢？这边举一个例子，一年前我曾经示范了怎么用语言模型玩文字冒险游戏，一年之前的玩法是这个样子，你就输入一段文字，告诉它说我们要来开始一个文字冒险游戏，然后由玩家来决定要采取的动作，然后游戏开始的时候请详述故事背景。游戏开始，你只要输入这些文字，这个 Chat GPT 呢，它就可以主持一个文字冒险游戏，它会提供选项让你选，根据你的选项，它就会有不同的剧情展开。它可能会这样回答你，你是一个探险家，然后你听说一个古代宝藏的传说，你要踏上这个旅程，寻找古代的宝藏，给你三个选项，来选不同的选项，会有不同的结局。因为玩这个文字冒险游戏呢，只有文字实在是太干了，所以在一年之前，为了让整个游戏不么干，我把文字冒险游戏的叙述丢到一个文字生图的 AI，当时用的是 MidJourney，让它产生游戏的插图。

但今天你其实是可以有不一样的玩法，你可以多加一句，每次你描述完场景之后，请根据你的描述产生一张图。它一样会开始架构一个文字冒险游戏的场景，它说你醒来在一个阴暗的森林中，你有一个背包，背包里面有一个地图，然后看起来像是这个森林的地图，你还有小手电筒，你还有旧的日记，上面写着寻找遗失之城等等。讲完这些描述之后。Chat GPT 就去调用 DALL-E 生成一个跟场景有关的图，看起来真的有个森林，有个背包，有手电筒，还有地图，跟它刚才的描述呢，是满一致的。接下来它一样会给你几个选项，问你要走左边的小径，还是继续往前走，还是走右侧的小径。这样这个文字冒险游戏呢，玩起来就更带感了。其实这个 GPT 4 还有很多其它它可以使用的工具啦。我们刚才讲的几个，比如说调用搜索引

擎，调用文字生图的 AI 等等，是它内建本来就可以调用的工具。如果你想要调用更多工具的话，有一个东西叫做 GPT 的插件，这个插件里面现在有收集了超过上千个工具。

你要用这些工具的时候呢，你就要进入一个叫做插件的模式，然后在个模式里面从上千个工具里面选三个工具在对话的时候进行使用。回答刚才讲的问题，语言模型是怎么使用这些工具的呢？不要忘了语言模型只会做一件事，就是文字接龙。所以当它使用工具的时候，它也是用文字接龙的方式在使用工具的。

> 语言模型使用工具也是通过文字接龙方式。

怎么用文字接龙的方式来使用工具呢？例如，假设你现在想问语言模型，我用五美金可以换多少新台币，语言模型想要回答你这个问题，所以开始做文字接龙，它接了“五美金可以换”这几个字以后，觉得信息量不太够，因为到底美金跟台币兑换的汇率应该是多少呢，这个兑换的汇率啊，应该是随时间变动的，所以应该去网络上查一下最新的兑换比例。它就会产生一个特殊的符号，这个特殊的符号代表调用工具，你可以想像说它先定义好一个符号，比如说一个你平常在对话的时候绝对不会用到的符号，那个符号只要一出现就代表调用工具。接下来呢，它会继续去做文字接龙，直到它产生另外一个特殊符号，代表结束使用工具为止。在调用工具到结束使用工具这两个符号之间的文字，就是操作工具的指令。至于操作工具的指令该长什么样子，这个是你要事先定好的。假设在括号之前的东西代表要使用哪一个工具，比如说搜索代表要使用搜索引擎，括号里面的内容代表你要给个工具的内容，如果是搜索引擎的话，就是你要拿去搜索的个关键字，就把美金台币兑换这几个关键字，丢到 Google 上面，Google 就会回传给你一些相关的信息。比如说你马上就可以看个网页，个网页上是写一美金等于 31.5 台币，把你从网页上搜索到的结果，当作是文字接龙的一部分，贴到刚才已经产生的句子后面，这些网络上搜索到的内容就是语言模型做文字接龙已经产生出来的结果，语言模型会根据这些网络上搜索到的内容，再继续去做文字接龙。比如说我已经知道一美金就是 31.5 台币，但是五美金到底兑换多少台币呢，而如果用文字接龙的方式来计算，恐怕非常容易算错，所以再调用一次工具产生一段文字，直到调用工具结束为止，这一段文字就是操控工具的指令，括号前面是计算机，代表要调用计算机，括号里面的内容，就是你要输入给计算机的内容，5 乘以 31.5，计算机按一下以后，得到 157.5。我们把计算机的输出当作是文字接龙，已经产生出来的结果，再继续去做文字接龙，这个时候语言模型就不需要自己再去做任何数学了，这边前面都已经接出来是 157.5，它根据这段文字继续去做文字接龙，它只要复制就好，它把 157.5 复制出来，得到答案是 157.5 元新台币，输出结束的符号代表生成结束。

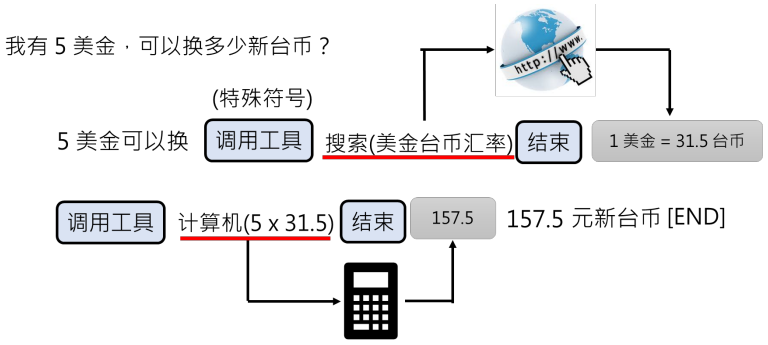


图 3.7　语言模型使用工具的过程