LLM Concepts Guide LLM概念指南

At their most basic level, Large Language Models (LLMs) are like sophisticated autocomplete. Given input text ("You can lead a horse to water,"), LLMs output text that's statistically likely to follow ("but you can't make it drink"), based on patterns learned from their training data. You can use this basic property of LLMs to power several different types of applications:

在最基本的层面上，大型语言模型（LLMs）就像是复杂的自动完成。给定输入文本（“你可以带马到水边，”），LLMs会输出基于其训练数据中学习到的模式，统计上可能跟随的文本（“但你不能强迫它喝水”）。您可以使用LLMs的这种基本属性来驱动几种不同类型的应用程序：

- Generate poetry, short stories, metaphors, blog posts, and other creative copy

  生成诗歌、短篇小说、隐喻、博客文章和其他创意文本
- Convert structured data to freeform text

  将结构化数据转换为自由形式的文本
- Extract and summarize information from freeform text

  从自由形式的文本中提取和总结信息
- Generate code 生成代码
- Translate 翻译
- Build a chatbot etc. 构建聊天机器人等。

Using LLMs to build these features doesn't require any Machine Learning (ML) expertise. You don't need to collect a large dataset or train a model. All it takes to start your first LLM prototype is to describe what you want the model to do in a few sentences.

使用LLMs构建这些功能不需要任何机器学习（ML）专业知识。您不需要收集大量数据集或训练模型。要启动您的第一个LLM原型，只需用几句话描述您希望模型执行的任务即可。

Example applications 示例应用

Poem Generator 诗歌生成器

User input: `Write me a four-line poem about puppies and Android phones. Make sure it rhymes.` 用户输入： `Write me a four-line poem about puppies and Android` `phones. Make sure it rhymes.`

Model output: 模型输出：

```
Puppies are cute,
Android phones are neat,
They both bring joy,
And make life complete.
```

List Generator 列表生成器

User input: **`Generate a bulleted list of items I need to pack for a three-day camping trip.`** 用户输入： `Generate a bulleted list of items I need to pack for a three-day camping trip.`

```
Model output:

*   Tent
*   Sleeping bag
*   Sleeping pad
*   Camp stove
*   Pots and pans
*   Utensils
*   Food
*   Water
*   First-aid kit
*   Sunscreen
*   Insect repellent
*   …
```

You can get LLMs to produce all sorts of useful behaviors like this, just by crafting the right input text, also called a **prompt**. The art and science of figuring out the right wording to get LLMs to do what you want is called **prompt design** (also called "prompt engineering" or simply "prompting").

您可以通过精心设计正确的输入文本（也称为提示）来让LLMs产生各种有用的行为，就像上面的例子一样。找出正确的措辞让LLMs执行您想要的操作的艺术和科学称为提示设计（也称为“提示工程”或简称“提示”）。

Prompt Design 101 提示设计101

The previous section showed some examples of prompts that contain an instruction, like 'Write me a poem'. This kind of instruction may work well for certain types of tasks. However, for other applications, another prompting technique called **few-shot prompting** may work better. Few-shot prompts take advantage of the fact that large language models are incredibly good at recognizing and replicating patterns in text data. The idea is to send the LLM a text pattern that it learns to complete. For example, let's say you want to build an application that takes as input a country name and outputs its capital city. Here's a text prompt designed to do just that:

前面的部分展示了一些包含指令的提示示例，例如“写一首诗”。这种指令可能对某些类型的任务有效。但是，对于其他应用程序，另一种提示技术称为few-shot提示可能更有效。Few-shot提示利用了大型语言模型在文本数据中识别和复制模式的能力。其想法是向LLM发送一个文本模式，它学会了如何完成。例如，假设您想构建一个应用程序，该应用程序以国家名称作为输入，并输出其首都城市。这是一个旨在实现此目的的文本提示：

```
Italy : Rome
France : Paris
Germany :
```

In this prompt, you establish a pattern: `[country] : [capital]`. If you send this prompt to a large language model, it will autocomplete the pattern and return something like this:

在此提示中，您建立了一个模式： `[country] : [capital]` 。如果将此提示发送到大型语言模型，则它将自动完成该模式并返回类似于以下内容：

```
     Berlin
Turkey : Ankara
Greece : Athens
```

This model response may look a little strange. The model returned not only the capital of Germany (the last country in your hand-written prompt), but also a whole list of additional country/capital pairs. That's because the LLM is "continuing the pattern." If all you're trying to do is build a function that tells you the capital of an input city ("Germany : Berlin"), you probably don't really care about any of the text the model generates after "Berlin." Indeed, as application designers, you'd probably want to truncate those extraneous examples. What's more, you'd probably want to **parameterize** the input, so that Germany is not a fixed string but a variable that the end user provides:

这个模型响应可能看起来有点奇怪。该模型不仅返回了德国的首都（手写提示中的最后一个国家），还返回了整个附加的国家/首都对列表。这是因为LLM正在“继续模式”。如果您要构建的所有内容都是一个函数，该函数告诉您输入城市的首都（“德国：柏林”），那么您可能并不关心模型在“柏林”之后生成的任何文本。实际上，作为应用程序设计者，您可能希望截断那些无关的示例。此外，您可能希望参数化输入，以便德国不是固定的字符串，而是最终用户提供的变量：

```
Italy : Rome
France : Paris
<user input here> :
```

You have just written a few-shot prompt for generating country capitals.

你刚刚写了一个用于生成国家首都的 few-shot 提示。

You can accomplish a large number of tasks by following this **few-shot prompt** template. Here's a few-shot prompt with a slightly different format that converts Python to Javascript:

通过遵循这个 few-shot prompt 模板，您可以完成大量任务。这是一个稍微不同格式的 few-shot prompt，可以将 Python 转换为 Javascript：

```
Convert Python to Javascript.
Python: print("hello world")
Javascript: console.log("hello world")
Python: for x in range(0, 100):
Javascript: for(var i = 0; i < 100; i++) {
Python: ${USER INPUT HERE}
Javascript:
```

Or, take this "reverse dictionary" prompt. Given a definition, it returns the word that fits that definition:

或者，接受这个“反向词典”提示。给定一个定义，它会返回符合该定义的单词：

```
Given a definition, return the word it defines.
Definition: When you're happy that other people are also sad.
Word: schadenfreude
Definition: existing purely in the mind, but not in physical reality
Word: abstract
```

```
Definition: ${USER INPUT HERE}
Word:
```

You might have noticed that the exact pattern of these few-shot prompts varies slightly. In addition to containing examples, providing instructions in your prompts is an additional strategy to consider when writing your own prompts, as it helps to communicate your intent to the model.

您可能已经注意到这些少样本提示的确切模式略有不同。除了包含示例外，在提示中提供说明是编写自己的提示时要考虑的另一种策略，因为它有助于向模型传达您的意图。

Prompting vs traditional software development

提示与传统软件开发

Unlike traditional software that's designed to a carefully written spec, the behavior of LLMs is largely opaque even to the model trainers. As a result, you often can't predict in advance what types of prompt structures will work best for a particular model. What's more, the behavior of an LLM is determined in large part by its training data, and since models are continually tuned on new datasets, sometimes the model changes enough that it inadvertently change which prompt structures work best. What does this mean for you? Experiment! Try different prompt formats.

与专门设计为精心编写的规范的传统软件不同，LLM的行为对于模型训练者来说很大程度上是不透明的。因此，您通常无法预测哪种提示结构对于特定模型最有效。更重要的是，LLM的行为在很大程度上取决于其训练数据，由于模型不断地在新数据集上进行调整，有时模型会发生足以无意中改变哪些提示结构最有效的变化。这对您意味着什么？实验！尝试不同的提示格式。

Model parameters 模型参数

Every prompt you send to the model includes parameter values that control how the model generates a response. The model can generate different results for different parameter values. The most common model parameters are:

您发送给模型的每个提示都包括控制模型生成响应的参数值。模型可以根据不同的参数值生成不同的结果。最常见的模型参数包括：

1. **Max output tokens:** Specifies the maximum number of tokens that can be generated in the response. A token is approximately four characters. 100 tokens correspond to roughly 60-80 words.

   最大输出标记数：指定响应中可以生成的标记的最大数量。一个标记大约是四个字符。100个标记大约对应60-80个单词。

2. **Temperature:** The temperature controls the degree of randomness in token selection. The temperature is used for sampling during response generation, which occurs when `topP` and `topK` are applied. Lower temperatures are good for prompts that require a more deterministic/less open-ended response, while higher temperatures can lead to more diverse or creative results. A temperature of 0 is deterministic, meaning that the highest probability response is always selected.

   温度：温度控制令牌选择中的随机程度。温度用于响应生成期间的抽样，当应用 `topP` 和 `topK` 时发生。较低的温度适用于需要更确定性/不太开放式响应的提示，而较高的温度可以导致更多样化或创造性的结果。温度为0是确定性的，这意味着始终选择最高概率响应。

3. **`topK`:** The `topK` parameter changes how the model selects tokens for output. A `topK` of 1 means the selected token is the most probable among all the tokens in the model's vocabulary (also called greedy decoding), while a `topK` of 3 means that the next token is selected from among the 3 most probable using the temperature. For each token selection step, the `topK` tokens with the highest probabilities are sampled. Tokens are then further filtered based on `topP` with the final token selected using temperature sampling.

   `topK` ： `topK` 参数改变了模型选择输出令牌的方式。1的 `topK` 意味着所选令牌是模型词汇表中所有令牌中最有可能的（也称为贪婪解码），而3的 `topK` 意味着下一个令牌是从使用温度最有可能的3个令牌中选择的。对于每个令牌选择步骤，将抽样最高概率的 `topK` 个令牌。然后根据 `topP` 进一步过滤令牌，使用温度抽样选择最终令牌。

4. **`topP`:** The `topP` parameter changes how the model selects tokens for output. Tokens are selected from the most to least probable until the sum of their probabilities equals the `topP` value. For example, if tokens A, B, and C have a probability of 0.3, 0.2, and 0.1 and the `topP` value is 0.5, then the model will select either A or B as the next token by using the temperature and exclude C as a candidate. The default `topP` value is 0.95.

   `topP` ： `topP` 参数会改变模型选择输出令牌的方式。令牌按照从最可能到最不可能的顺序选择，直到它们的概率之和等于 `topP` 值。例如，如果令牌 A、B 和 C 的概率分别为 0.3、

0.2 和 0.1， `topP` 值为 0.5，则模型将使用温度选择 A 或 B 作为下一个令牌，并将 C 排除在候选之外。默认的 `topP` 值为 0.95。

Types of prompts 提示类型

Depending on the level of contextual information contained in them, prompts are broadly classified into three types.

根据它们所包含的上下文信息的程度，提示被广泛地分为三种类型。

Zero-shot prompts 零样本提示

These prompts do not contain examples for the model to replicate. Zero-shot prompts essentially show the model's ability to complete the prompt without any additional examples or information. It means the model has to rely on its pre-existing knowledge to generate a plausible answer.

这些提示不包含模型复制的示例。零-shot提示基本上展示了模型在没有任何额外示例或信息的情况下完成提示的能力。这意味着模型必须依靠其现有的知识来生成一个合理的答案。

Some commonly used zero-shot prompt patterns are:

一些常用的零-shot提示模式包括：

- Instruction-content 指令内容

```
<Overall instruction>
<Content to operate on>
```

For example, 例如，

```
Summarize the following into two sentences at the third-grade level:

Hummingbirds are the smallest birds in the world, and they are also one of the
most fascinating. They are found in North and South America, and they are known
for their long, thin beaks and their ability to fly at high speeds.

Hummingbirds are made up of three main parts: the head, the body, and the tail.
```

```
The head is small and round, and it contains the eyes, the beak, and the brain.
The body is long and slender, and it contains the wings, the legs, and the
heart. The tail is long and forked, and it helps the hummingbird to balance
while it is flying.

Hummingbirds are also known for their coloration. They come in a variety of
colors, including green, blue, red, and purple. Some hummingbirds are even able
to change their color!

Hummingbirds are very active creatures. They spend most of their time flying,
and they are also very good at hovering. Hummingbirds need to eat a lot of food
in order to maintain their energy, and they often visit flowers to drink nectar.

Hummingbirds are amazing creatures. They are small, but they are also very
powerful. They are beautiful, and they are very important to the ecosystem.
```

- Instruction-content-instruction 指令-内容-指令

```
<Overall instruction or context setting>
<Content to operate on>
<Final instruction>
```

For example, 例如，

```
Here is some text I'd like you to summarize:

Summarize the following into two sentences at the third-grade level:

Hummingbirds are the smallest birds in the world, and they are also one of the
most fascinating. They are found in North and South America, and they are known
for their long, thin beaks and their ability to fly at high speeds. Hummingbirds
are made up of three main parts: the head, the body, and the tail. The head is
small and round, and it contains the eyes, the beak, and the brain. The body is
long and slender, and it contains the wings, the legs, and the heart. The tail
is long and forked, and it helps the hummingbird to balance while it is flying.
Hummingbirds are also known for their coloration. They come in a variety of
colors, including green, blue, red, and purple. Some hummingbirds are even able
to change their color! Hummingbirds are very active creatures. They spend most
of their time flying, and they are also very good at hovering. Hummingbirds need
to eat a lot of food in order to maintain their energy, and they often visit
flowers to drink nectar. Hummingbirds are amazing creatures. They are small, but
```

```
they are also very powerful. They are beautiful, and they are very important to
the ecosystem.

Summarize it in two sentences at the third-grade reading level.
```

- Continuation. Sometimes, you can have the model continue text without any instructions. For example, here is a zero-shot prompt where the model is intended to continue the input provided:

  继续。有时，您可以让模型在没有任何指令的情况下继续文本。例如，这是一个零-shot提示，模型旨在继续提供的输入：

```
Once upon a time, there was a little sparrow building a nest in a farmer's
barn. This sparrow
```

Use zero-shot prompts to generate creative text formats, such as poems, code, scripts, musical pieces, email, letters, etc.

使用零-shot提示生成创意文本格式，例如诗歌、代码、脚本、音乐作品、电子邮件、信件等。

One-shot prompts 一次性提示

These prompts provide the model with a single example to replicate and continue the pattern. This allows for the generation of predictable responses from the model.

这些提示为模型提供了一个单一的示例来复制和继续模式。这允许模型生成可预测的响应。

For example, you can generate food pairings like:

例如，您可以生成食品搭配，如：

```
Food: Apple
Pairs with: Cheese
Food: Pear
Pairs with:
```

Few-shot prompts 少样本提示

These prompts provide the model with multiple examples to replicate. Use few-shot prompts to complete complicated tasks, such as synthesizing data based on a pattern.

这些提示为模型提供了多个示例来复制。使用少量提示来完成复杂的任务，例如基于模式合成数据。

An example prompt may be: 一个示例提示可能是：

```
Generate a grocery shopping list for a week for one person. Use the JSON format
given below.
{"item": "eggs", "quantity": "6"}
{"artist": "bread", "quantity": "one loaf"}
```

LLMs under the hood LLM底层原理

This section aims to answer the question - ***Is there randomness in LLMs' responses, or are they deterministic?***

本节旨在回答一个问题——LLM的回答中是否存在随机性，或者它们是确定性的？

The short answer - yes to both. When you prompt an LLM, its text response is generated in two stages. In the first stage, the LLM processes the input prompt and generates a **probability distribution** over possible tokens (words) that are likely to come next. For example, if you prompt with the input text “The dog jumped over the ... ”, the LLM will produce an array of probable next words:

简短的回答是——两者都有。当您提示LLM时，它的文本回答是通过两个阶段生成的。在第一阶段，LLM处理输入提示并生成一个可能的令牌（单词）概率分布，这些单词可能是接下来出现的。例如，如果您提示输入文本“狗跳过了......”，LLM将生成一个可能的下一个单词数组：

```
[("fence", 0.77), ("ledge", 0.12), ("blanket", 0.03), …]
```

This process is deterministic; an LLM will produce this same distribution every time it's input the same prompt text.

这个过程是确定性的；每次输入相同的提示文本，LLM都会产生相同的分布。