# 数据概览

在当代，大量的大型语言模型（LLM）应用需要用户特定的数据，而这部分数据并不包含在模型的训练集中。因此，如何加载、转化、存储和查询这些数据，便成为我们需要解决的核心问题。针对这一需求，LangChain提供了一整套解决方案。它通过一系列的构建模块，包括文档加载器（Document loaders）、文档转换器（Document transformers）、文本嵌入模型（Text embedding models）、向量存储（Vector stores）以及检索器（Retrievers），让我们能够自由地与自己的数据和大型语言模型进行交互。

在以下的篇章中，我们将详细地探讨每一个构建模块的实现原理。

#### 2. 文档加载器：灵活加载文档的利器

文档加载器能够从众多不同的来源加载文档。一个文档可以简单地理解为一段文字和其相关的元数据。例如，文档加载器可以从一个简单的.txt文件中，从任意网页的文本内容，甚至从YouTube视频的字幕中加载数据。

文档加载器提供一个”load”方法，用于从配置的数据源加载数据作为文档。它们还可选择性地实现”lazy load”，以便于懒加载数据到内存中。

#### 3. 文档转换器：为你的应用精确定制文档

一旦文档被加载，我们往往需要对它们进行一定的转换，以便更好地适应我们的应用。最简单的例子是，我们可能需要将一个长文档分割成更小的片段，以便它们能够适应模型的上下文窗口。LangChain内置了多种文档转换器，使得分割、组合、过滤和其他形式的文档操作变得轻而易举。

#### 4. 文本嵌入模型：将无结构文本转化为浮点数列表

Embeddings类是为接口化文本嵌入模型而设计的一个类。目前有许多嵌入模型提供商（如OpenAI, Cohere, Hugging Face等），而这个类就是为了提供一个所有这些供应商的标准接口。

Embeddings类将一段文本创建为一个向量表示。这是有用的，因为我们可以将文本在向量空间中进行思考，并做诸如语义搜索这样的操作，我们可以寻找在向量空间中最相似的文本片段。

LangChain中的基础Embeddings类公开了两种方法：一种用于嵌入文档，另一种用于嵌入查询。前者接受多个文本作为输入，而后者接受单个文本。之所以将这两种方法分开，是因为一些嵌入提供商对待待查询的文档与查询本身有着不同的嵌入方法。

#### 5. 向量存储：为你的嵌入数据提供存储和搜索功能

最常见的存储和搜索无结构数据的方式之一是将其嵌入并存储生成的嵌入向量，然后在查询时嵌入无结构查询并检索那些与嵌入查询“最相似”的嵌入向量。向量存储负责存储嵌入数据并执行向量搜索。

#### 6. 检索器：返回文档的接口

检索器是一个能够根据无结构查询返回文档的接口。它比向量存储的应用更为广泛。检索器不需要能够存储文档，只需返回（或检索）文档即可。向量存储可以作为检索器的支撑结构，但也存在其他类型的检索器。

以上就是LangChain通过这些技术准备数据，让我们使用自己的数据与大型语言模型交互的全过程。LangChain用这种简单而高效的方式，让我们可以轻松地处理各种数据，并使之与我们的大型语言模型交互。