# LangChain的文档转换器和文本分割器：工作原理与应用

LangChain为处理语言数据提供了一系列内置工具，包括文档加载器、文档转换器和文本分割器等。在您加载了文档之后，通常需要对其进行转换以更好地适应您的应用。这就需要用到LangChain的文档转换器和文本分割器。

#### 1. 文档转换器

文档转换器可以轻松地将文档进行分割、合并、过滤和其他操作，以满足您的实际需求。例如，您可能希望将长文档分割成小块，以便适应您模型的上下文窗口。

#### 2. 文本分割器

当处理长文本时，往往需要将文本分割成块。尽管这看起来简单，但实际上可能涉及很多复杂性。理想情况下，您会希望将语义相关的文本部分保持在一起。而”语义相关”的含义可能取决于文本的类型。下面将介绍几种实现这一目标的方法。

在高层次上，文本分割器的工作原理如下：

* 将文本分割成小的、语义上有意义的块（通常是句子）。
* 将这些小块开始组合成一个大的块，直到达到某个大小（通过某种函数进行测量）。
* 一旦达到该大小，将该块作为自己的文本片段，然后开始创建新的文本块，新的文本块和前一个文本块会有一些重叠（以保持块与块之间的上下文）。

这意味着，您可以沿着两个不同的轴来定制您的文本分割器：

* 文本如何被分割
* 块的大小如何被测量

#### 3. 使用文本分割器

默认推荐的文本分割器是RecursiveCharacterTextSplitter。这个文本分割器接受一个字符列表，它尝试基于第一个字符进行分割，但如果任何块太大，它就会移动到下一个字符，依此类推。默认情况下，它尝试分割的字符是[“”, “”, ” “,””]。

除了可以控制分割的字符，您还可以控制以下几点：

* length\_function：如何计算块的长度。默认只计算字符数量，但是在此通常会传入一个标记计数器。
* chunk\_size：您的块的最大大小（由长度函数测量）。
* chunk\_overlap：块之间的最大重叠。有一些重叠可以在块之间保持连续性（例如采用滑动窗口的方式）。
* add\_start\_index：是否在元数据中包含每个块在原始文档中的起始位置。

通过以上内容，我们可以看到LangChain的文档转换器和文本分割器为处理和转换大规模文本提供了有效的工具，无论是文本的分割、组合、过滤还是其他操作，都能够得心应手。