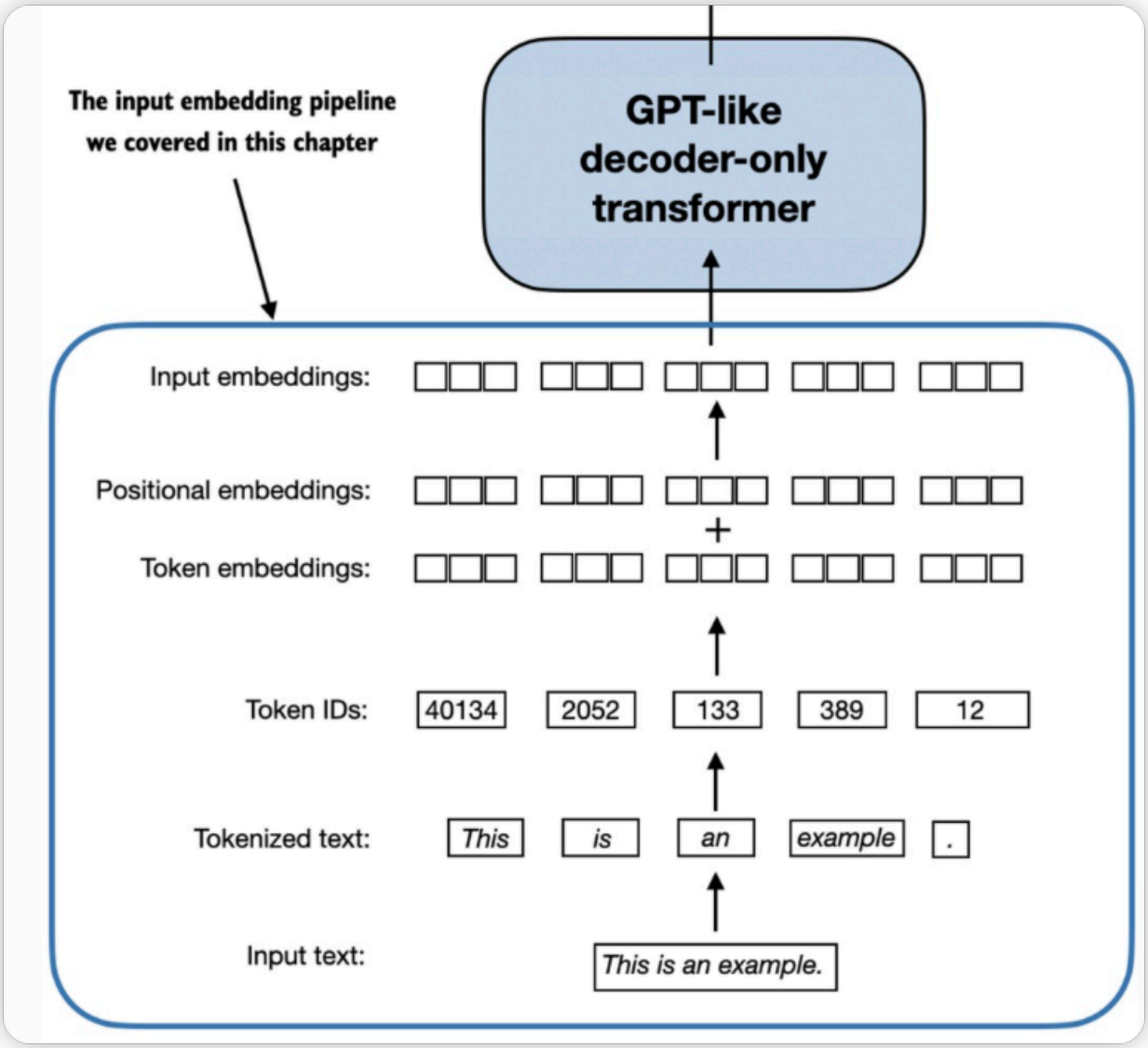
Input 转 Input embedding

- 嵌入 embedding：将特征数据转换为向量格式。步骤如下：



下面依次分析从input text 到 input embeddings的过程。

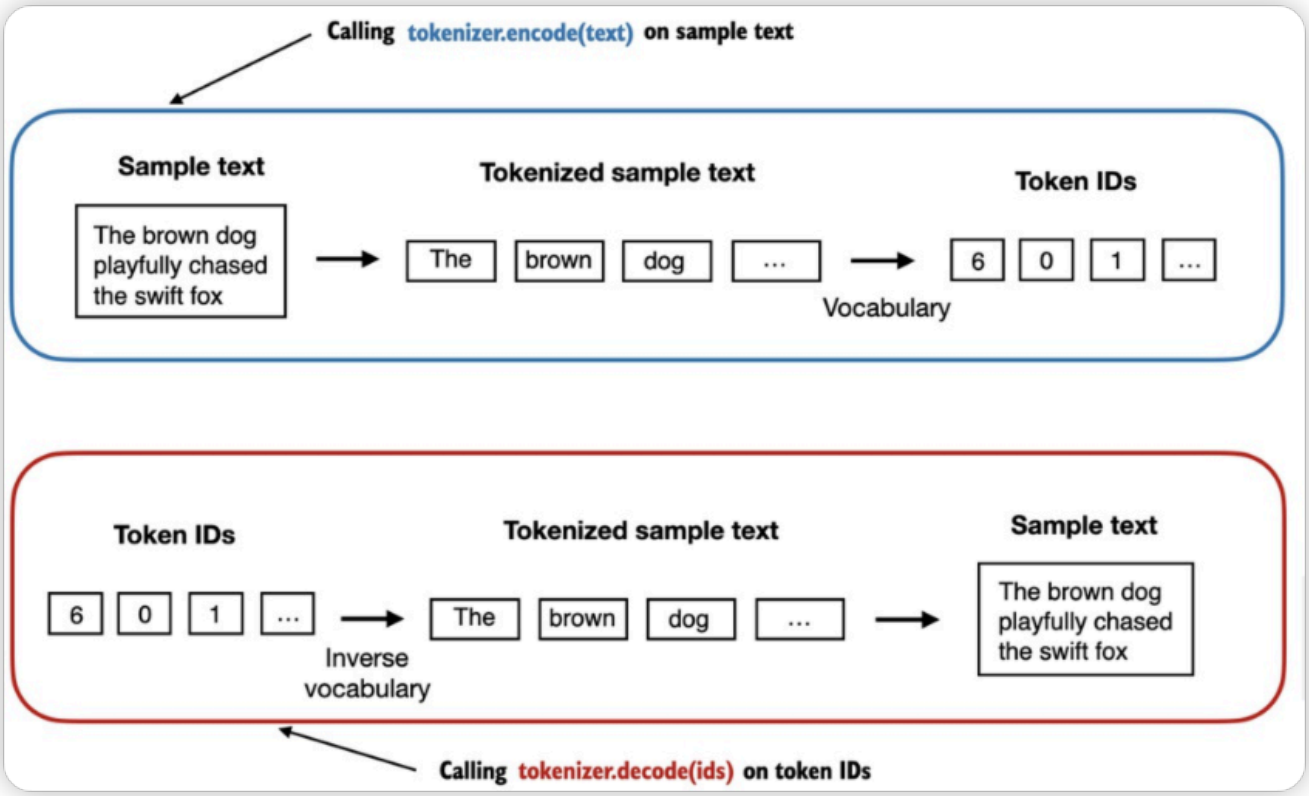
Input_text 转 Tokenized text：分词

移除空格可以减少内存和计算需求；
保留空格在我们训练对文本的精确结构敏感的模型时可能是有用的；

```
# 一个简单有效的分词方案：
import re
preprocessed = re.split(r'[,.?_!"()\']|--|\s', raw_text)

# 先遍历每一个item，再做strip()操作，最后判断不为空加入list
preprocessed = [item.strip() for item in preprocessed if
item.strip()]
```

Tokenized text 转 Token IDs

构建一个包含所有token的词汇表，和一个逆向版本（LLM输出从数字转换回文本）。



特殊上下文token

- <|UNK|>：不属于词汇表的单词
- <|endoftext|>：标记特定段落的开始和结束

还有一些研究人员会考虑的其他的特殊token：

- <|BOS|>（beginning of sequence）：此标记标记文本的开始。LLM表示一段内容开始的位置。
- <|EOS|>（end of sequence）：这个标记位于文本的末尾，在连接多个不相关的文本时特别有用，类似于<|内文|>.例如，当合并两个不同的维基百科文章或书籍时，<|EOS|>令牌指示一篇文章结束的位置和下一篇文章开始的位置。
- <|PAD|>（padding）：当训练批量大小大于1的LLMs时，该批可能包含不同长度的文本。为了确保所有文本具有相同的长度，使用<|PAD|>标记扩展或“填充”较短的文本，直到批次中最长文本的长度。

字节对编码

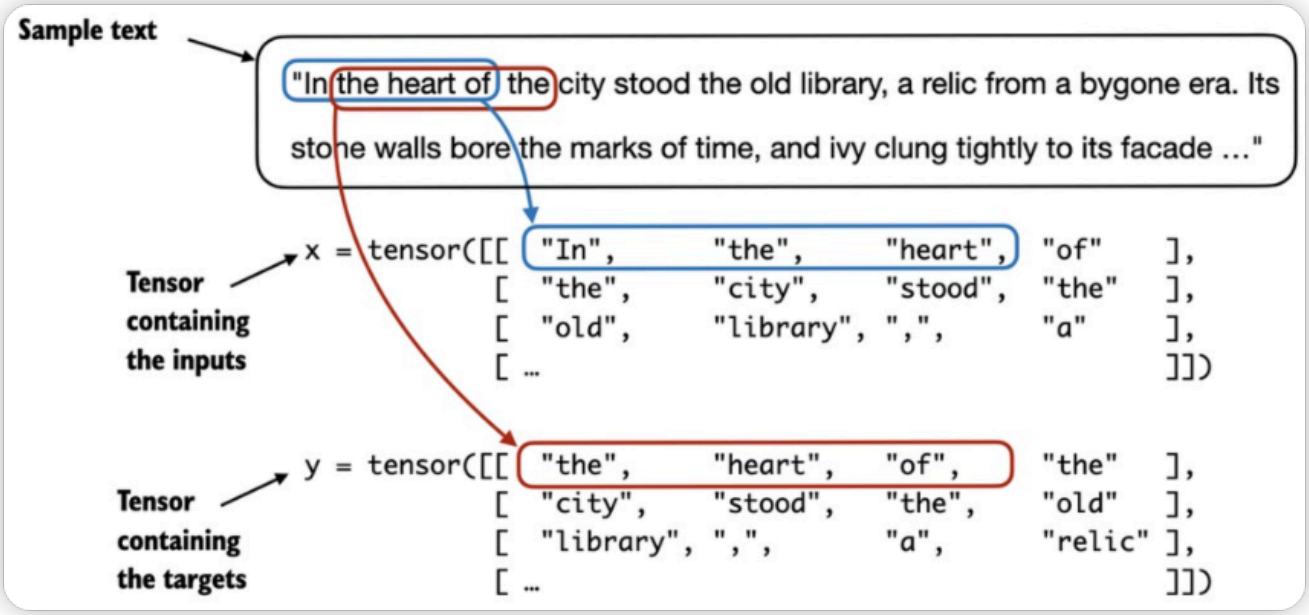
算法较复杂，直接使用了 `tiktoken` 库来。

```
import tiktoken

# 编码
tokenizer = tiktoken.get_encoding("gpt2")
text = "Hello, do you like tea? <|endoftext|> In the sunlit
terraces of some"
integers = tokenizer.encode(text, allowed_special={"
<|endoftext|>"})
print(integers)

# 解码
strings = tokenizer.decode(integers)
print(strings)
```

BPE使用的算法会将不在预定义词表里的单词分解为更小的子单词单元或者甚至是独立的字母，使BPE可以处理词表外的单词。

Token IDs 转 Token embeddings：得到输入-目标对

使用滑动窗口进行数据采样



> 这里窗口大小（即上下文大小content_size） `max_length` = 4，
> 步长 `stride` = 1。

构建词嵌入：特征可计算

> 数据处理的最后一步

token本身无法计算，将其映射到一个连续的向量空间，才可进行后续运算。独热编码（

- 使非偏序关系的变量取值不具有偏序性，并且到圆点是等距的。
- 将离散特征的取值扩展到了欧式空间，让特征之间的**距离计算更加合理**

）+ 矩阵方法的高效实现。

exp：

```
vocab_size = 6 # 词汇表大小
output_dim = 3 # 编码大小

# 设计随机种子，确保Embedding实验结果的可重复性
torch.manual_seed(123)
```

```
token_embedding_layer = torch.nn.Embedding(vocab_size, output_dim)

...

token_embeddings = token_embedding_layer(inputs)
```

token_embedding_layer：

```
Parameter containing:
tensor([[ 0.3374, -0.1778, -0.1690],
                [ 0.9178, 1.5810, 1.3010],
                [ 1.2753, -0.2010, -0.1606],
                [-0.4015, 0.9666, -1.1481],
                [-1.1589, 0.3255, -0.6315],
                [-2.8400, -0.7849, -1.4096]], requires_grad=True)
```

词汇表共6个词汇，1对应 `[ 0.3374, -0.1778, -0.1690]`，2对应 `[ 0.9178, 1.5810, 1.3010]`，...，6对应 `[-2.8400, -0.7849, -1.4096]`。方便进行后续运算

Positonal embeddings：词位置编码

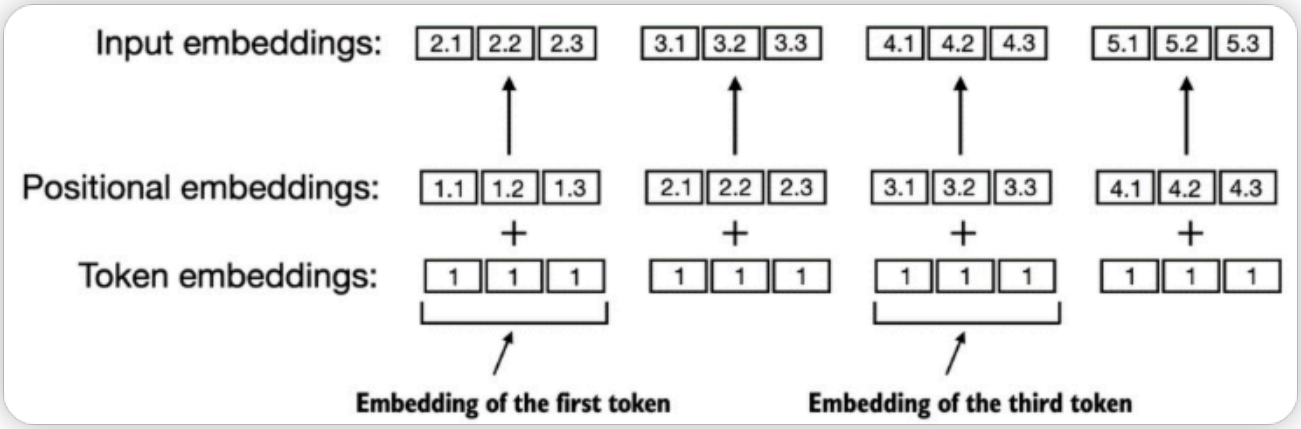
> 引入：自注意力机制本身也不关注位置，因此需将额外的位置信息注入的LLM中

相对位置编码

彼此之间有多远：即使模型在训练中没有看到这样的长度，可也更好推广。

绝对位置编码

在哪个确切位置。OpenAI的GPT使用绝对位置编码。


Input embeddings:
2.1 2.2 2.3
3.1 3.2 3.3
4.1 4.2 4.3
5.1 5.2 5.3
Positional embeddings:
1.1 1.2 1.3
2.1 2.2 2.3
3.1 3.2 3.3
4.1 4.2 4.3
+
Token embeddings:
1 1 1
1 1 1
1 1 1
1 1 1
Embedding of the first token
Embedding of the third token