

# Tokenization学习笔记

## 1.基础流程：

1.文本预处理（如删去HTML，标点符号，进行大小写转换，去除停用词（如the,is,and等无意义词）等）

2.分词处理（Tokenization）：

- (1) 分词：将文本转换为Token
- (2) 转索引：将Token映射到词汇表中的索引
- (3) 填充：对齐不同长度的句子
- (4) 生成Attention Mask：标记哪些Token需要关注

## BERT Tokenizer：

在本题中，我使用的是BERT tokenizer，这是一种基于Wordpiece方法的分词器，用于将文本转化成BERT模型可以处理的格式。它将输入的文本转换为一系列的 **token**（词或子词），并将其映射到预训练的BERT模型的词汇表中。

### WordPiece:

WordPiece 的目标是将文本拆分成子词（subwords）或更小的单元，避免将未知词汇（OOV，Out-of-Vocabulary）作为一个整体单词处理。

1.词汇表初始化：

WordPiece 会从最常见的词开始，并通过统计训练语料中的词频来构建词汇表。初始的词汇表会包含一些常见的词，特殊标记（如 `[CLS]`, `[SEP]`）和常见子词单元。【`[CLS]` 位于序列的开头，用于表示整个文本的特征。`[SEP]` 用于分隔句子或标记句子的结束。】例如：

```
[CLS] I love this movie! [SEP]
```

2.分解单词为子词：

对于一个没有出现在词汇表中的单词，WordPiece 会将它拆分成更小的部分，直到每部分都在词汇表中。比如，词汇表中没有 "unhappiness"，但词汇表中可能包含 "un", "happiness"。那么，"unhappiness" 就会被拆解为 "un" 和 "happiness"

3.词汇表的构建：

WordPiece 的词汇表通过 **最大化数据压缩** 和 **最小化词汇表大小** 的平衡来构建。这样既能处理未登录词，又能保证词汇表不会过大

## 基本流程：

### 1.文本分词：

**BERT Tokenizer** 会将输入文本分解成子词（subword），并将它们映射到 BERT 模型词汇表中的索引。

## 2. 特殊符号的添加：

**[CLS]**: BERT 特别设计了一个起始符号 `[CLS]`，它用作每个输入序列的开始，并且在训练中常用来做分类任务的目标。

**[SEP]**: 这个符号用于分隔不同的句子或者文本段落。如果输入的是单个句子，它通常是文本的结尾。

## 3. 分词后的文本转换为token IDs:

在实际操作中，`BertTokenizer` 会将文本转换成一系列整数 (token IDs)，这些整数是通过 BERT 词典表中的索引查找来的。

## 4. 填充 (Padding) 与截断 (Truncation) :

**Padding**: 如果输入的文本长度小于模型规定的最大长度 (通常是 512)，则会进行 **填充**，即在句子后面填充特殊符号 `[PAD]`，直到达到指定的最大长度。

**Truncation**: 如果输入的文本长度超过了最大长度，则会 **截断** 多余的部分，通常会截掉从序列末尾开始的部分。

## 5. Attention Mask:

生成的 **attention mask** 是一个与输入序列相同长度的二进制向量。值为 1 表示该位置的 token 是有效的 (即它是实际的文本数据)，值为 0 表示该位置是填充部分。模型会根据 attention mask 来避免计算填充部分。