

# 03

## 基础组件之Tokenizer

- (1) Tokenizer简介
- (2) Tokenizer基本使用方法
- (3) Fast / Slow Tokenizer

## Tokenizer 简介

- **数据预处理**

- **Step1 分词**: 使用分词器对文本数据进行分词（字、字词）；
- **Step2 构建词典**: 根据数据集分词的结果，构建词典映射（这一步并不绝对，如果采用预训练词向量，词典映射要根据词向量文件进行处理）；
- **Step3 数据转换**: 根据构建好的词典，将分词处理后的数据做映射，将文本序列转换为数字序列；
- **Step4 数据填充与截断**: 在以batch输入到模型的方式中，需要对过短的数据进行填充，过长的数据进行截断，保证数据长度符合模型能接受的范围，同时batch内的数据维度大小一致。

**现在: Tokenizer is all you need!**

# 基础组件之Tokenizer


## Tokenizer 基本使用

- 加载保存 (from\_pretrained / save\_pretrained)
- 句子分词 (tokenize)
- 查看词典 (vocab)
- 索引转换 (convert\_tokens\_to\_ids / convert\_ids\_to\_tokens)
- 填充截断 (padding / truncation)
- 其他输入 (attention\_mask / token\_type\_ids)

# 基础组件之Tokenizer

## Tokenizer 基本使用

- 加载保存 (from\_pretrained / save\_pretrained)
- 句子分词 (tokenize)
- 查看词典 (vocab)
- 索引转换 (convert\_tokens\_to\_ids / convert\_ids\_to\_tokens)
- 填充截断 (padding / truncation)
- 其他输入 (attention\_mask / token\_type\_ids)



tokenizer(inputs)

# 基础组件之Tokenizer

## Fast / Slow Tokenizer

- **FastTokenizer**

- 基于Rust实现, 速度快
- offsets\_mapping、word\_ids

- **SlowTokenizer**

- 基于Python实现, 速度慢

```
%%time
# 单条循环处理
for i in range(10000):
    fast_tokenizer(sen)
✓ 0.3s
```

CPU times: total: 78.1 ms  
Wall time: 312 ms

```
%%time
# 单条循环处理
for i in range(10000):
    slow_tokenizer(sen)
✓ 0.8s
```

CPU times: total: 281 ms  
Wall time: 872 ms

```
%%time
# 处理batch数据
res = fast_tokenizer([sen] * 10000)
✓ 0.1s
```

CPU times: total: 359 ms  
Wall time: 87.1 ms

```
%%time
# 处理batch数据
res = slow_tokenizer([sen] * 10000)
✓ 0.7s
```

CPU times: total: 188 ms  
Wall time: 705 ms