Tokenizer分词器

<https://huggingface.co/docs/transformers/tokenizer_summary>

**加载Bert分词器**

from transformers import BertTokenizer

tokenizer = BertTokenizer.from\_pretrained("bert-base-cased")

**加载轻量化Bert分词器**

from transformers import DistilBertTokenizerFast

tokenizer = DistilBertTokenizerFast.from\_pretrained('distilbert-base-cased')

**加载预训练的分词器**

tokenizer = AutoTokenizer.from\_pretrained(模型名)

#加载Bert: "bert-base-cased"

#加载GPT2："gpt2"

**利用已有的分词器训练新分词器**(需要自定义text\_batch\_iterator)

new\_tokenizer = tokenizer.train\_new\_from\_iterator(text\_batch\_iterator(), vocab\_size=词典大小)

**进行文本分词**

tokenizer.tokenize("text") #直接分词(仅一条text)

tokenizer([“text1”,”text2”])[ 'input\_ids'] #分词后映射为词ID(多条text)

**获取各词的ID**

word\_id\_list = tokenizer.convert\_tokens\_to\_ids(token)

**获取ID对应的词**

decoded\_string = tokenizer.decode(word\_id\_list)

**保存分词器**

tokenizer.save\_pretrained(folder\_path)

#会在folder\_path下保存tokenizer\_config.json, special\_tokens\_map.json, vocab.json， merges.txt, added\_tokens.json, tokenizer.json六个文件

搭建Bert Tokenizer

[..\..\..\..\..\..\GeorgeTown\FALL2022\ANLY-580\Lab\Lab8 搭建Bert,GPT-2,Albert Tokenizer\lab-08.html](../../../../../../GeorgeTown/FALL2022/ANLY-580/Lab/Lab8%20搭建Bert,GPT-2,Albert%20Tokenizer/lab-08.html)

**Bert Tokenizer组件：**

**normalizer,pre\_tokenizer,model,post\_processor,decoder**

from tokenizers import decoders, models, processors, trainers

**初始化分词器**

from tokenizers import Tokenizer

tokenizer = Tokenizer(models.WordPiece(unl\_token="[UNK]"))

#unl\_token为对text中所含的词典中没有的词的token

## normalizer：文本清洗

**使用Bert自带的normalizer**

from tokenizers import normalizers

tokenizer.normalizer = normalizers.BertNormalizer(lowercase=True)

#Bert的预处理是转小写

**自定义分词器normalizer pipeline**

from tokenizers import normalizers

tokenizer.normalizer = normalizers.Sequence(

[normalizers.NFD(), normalizers.Lowercase(), normalizers.StripAccents(),…]

)

## pre\_tokenizer：分词器

**使用Bert自带的pre\_tokenizer**

from tokenizers import pre\_tokenizers

tokenizer.pre\_tokenizer = pre\_tokenizers.BertPreTokenizer()

**测试pre-tokenizer功能**

tokenizer.pre\_tokenizer.pre\_tokenize\_str("test\_text")

## model: 训练Tokenizer对subword的表示

**配置训练器**

special\_tokens = ["[UNK]", "[PAD]", "[CLS]", "[SEP]", "[MASK]"] #Bert所用特殊token

trainer = trainers.WordPieceTrainer(vocab\_size=字典大小, special\_tokens=special\_tokens)

**使用iterator来传入数据**(需要自定义data\_batch\_iterator,适用于大数据**)**

tokenizer.train\_from\_iterator(data\_batch\_iterator(), trainer=trainer)

**通过读取文件来传入数据**

tokenizer.train\_from\_iterator(file, trainer=trainer)

## Post\_processor: 加入模型所需特殊token

#Bert中为插入[CLS]和[SEP]

cls\_token\_id = tokenizer.token\_to\_id("[CLS]")

sep\_token\_id = tokenizer.token\_to\_id("[SEP]")

在模板中指出如何用一句话($A)或两句话($A和$B)组织特殊的令牌。后跟的数字表示要给每个部分的令牌类型ID。

tokenizer.post\_processor = processors.TemplateProcessing(

single=f"[CLS]:0 $A:0 [SEP]:0",

pair=f"[CLS]:0 $A:0 [SEP]:0 $B:1 [SEP]:1",

special\_tokens=[

("[CLS]", cls\_token\_id),

("[SEP]", sep\_token\_id),

],

)

## Decoder

tokenizer.decoder = decoders.WordPiece(prefix="##")

#Bert特殊前缀##, 并采用单词级encode和decode

## 进行encode

encoding = tokenizer.encode(text1,text2,…)

**查看文本的Bert Tokenizer encoding结果**

encoding.tokens

**查看文本的Bert Tokenizer encoding令牌分配**

encoding.type\_ids

## 将Tokenizer植入Transformer类

from transformers import BertTokenizerFast

new\_tokenizer = BertTokenizerFast(tokenizer\_object=tokenizer)