华为“智能基座”系列课程

基于Mindspore的

Bert模型微调与推理实验

版本：1.0



华为技术有限公司

|  |
| --- |
| 版权所有 © 华为技术有限公司 2025。 保留一切权利。  非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。  商标声明  HW_POS_RBG_Vertical-150ppi 和其他华为商标均为华为技术有限公司的商标。  本文档提及的其他所有商标或注册商标，由各自的所有人拥有。  注意  您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或暗示的声明或保证。  由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。 |

|  |  |
| --- | --- |
| 华为技术有限公司 | |
| 地址： | 深圳市龙岗区坂田华为总部办公楼 邮编：518129 |
| 网址： | http://[e](http://e.huawei.com/).huawei.com |

目录

[1 华为云ModelArts环境创建 2](#_Toc181712417)

[1.1 创建ModelArts开发环境 2](#_Toc181712418)

[1.2 关闭/删除环境 5](#_Toc181712419)

[1.2.1 关闭环境 5](#_Toc181712420)

[1.2.2 删除环境 5](#_Toc181712421)

[2 基于MindSpore的Bert微调及推理实验 6](#_Toc181712422)

[2.1 模型简介 6](#_Toc181712423)

[2.2 实验环境准备 6](#_Toc181712424)

[2.3 实验过程 7](#_Toc181712425)

[2.4 模型推理 11](#_Toc181712426)

# 华为云ModelArts环境创建

在华为云ModelArts平台上创建AI框架为Mindspore-2.2、硬件环境为昇腾处理器+ARM的开发环境。

## 创建ModelArts开发环境

进入华为云ModelArts控制台

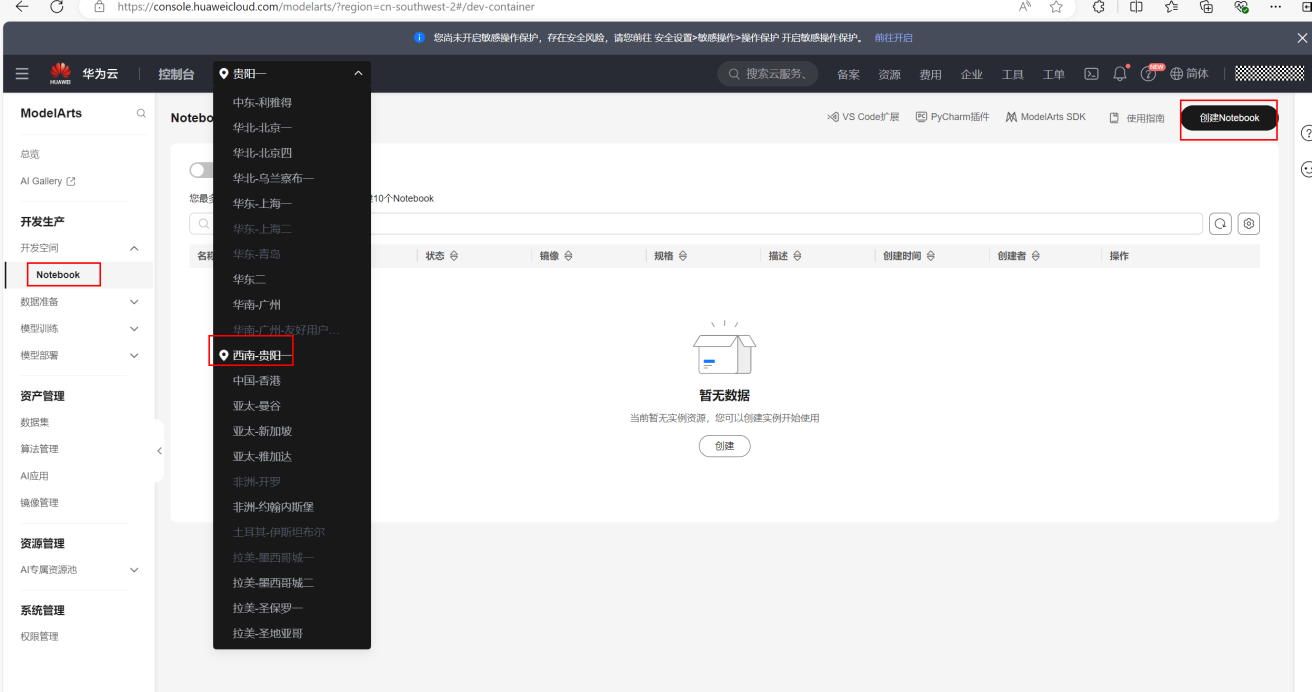
在[华为云ModelArts主页](https://www.huaweicloud.com/product/modelarts.html)，点击“管理控制台”进入ModelArts的管理页面。



华为云ModelArts主页

创建Notebook训练作业

控制台区域选择“西南-贵阳一”，在左侧菜单栏中选择“开发空间”的“Notebook”，如下图：



ModelArts控制台

在网页右上角，点击“创建Notebook”按钮，创建一个新的Notebook，其配置如下：

* 名称：自定义。
* **自动停止：打开，默认1小时。**
* 镜像：选择“公共镜像”，选择MindSpore2.2版本、Ascend+ARM的基础镜像。
* 类型：ASCEND。
* 规格：Ascend: 1\*ascend-snt9b(32G)|ARM: 24核 192GB。
* 存储配置：云硬盘EVS，128GB，可根据实验需要自定义。

注意：云硬盘从Notebook实例创建成功起，直至删除成功，每GB按照规定费用收费。

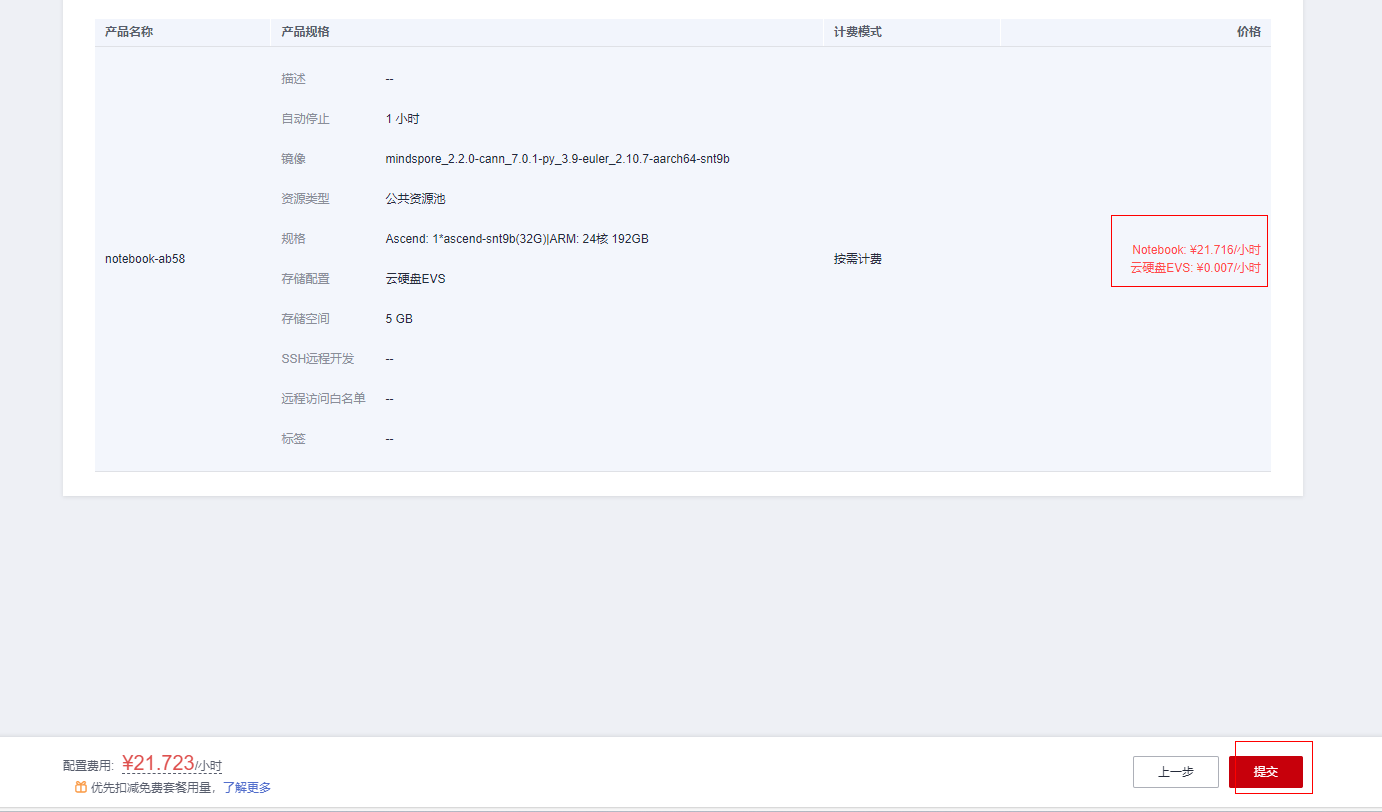
如图所示：





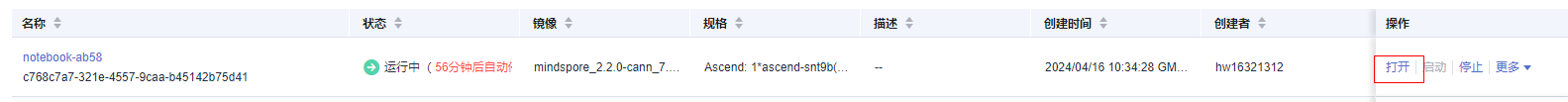
Notebook创建配置

配置完成之后“立即创建”，规格确认无误之后“提交”。



启动Notebook进入开发环境

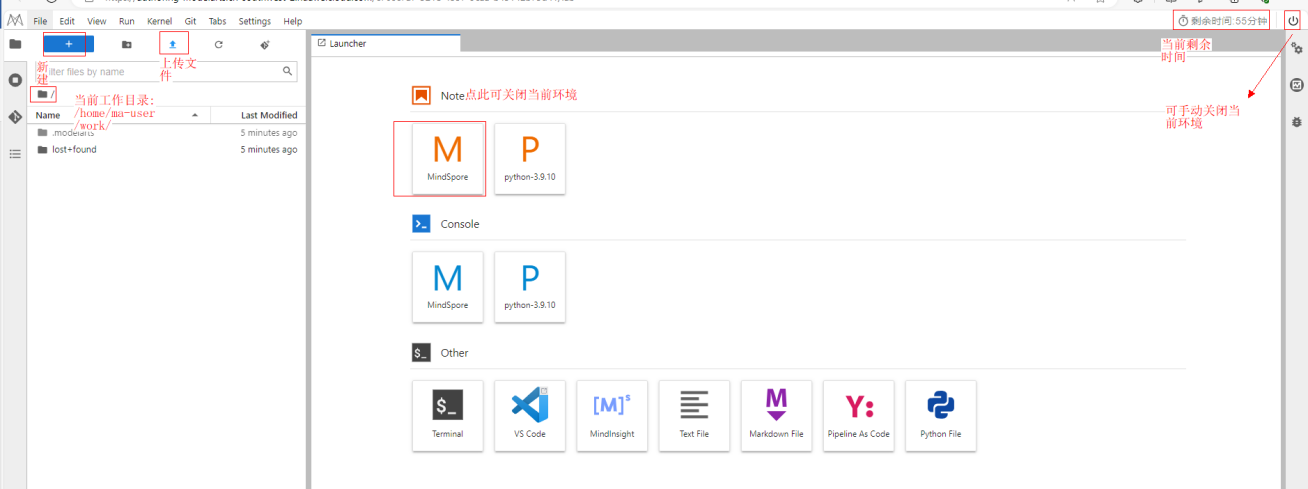
当上一步创建好Notebook状态显示为“运行中”时，在右侧操作中“打开”，即可进入在线编程页面。



Notebook实例入口

可以在此页面创建或编辑MindSpore的项目，如图所示：

\*注意：Notebook环境内上传、创建和编辑的文件均在/home/ma-user/work目录下。



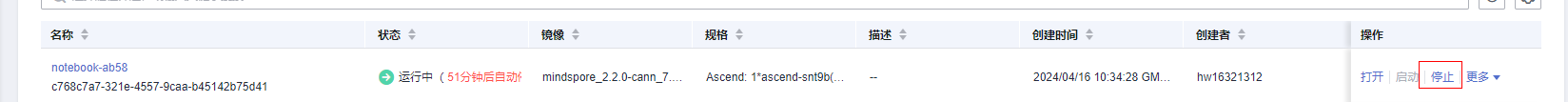
环境开发页面

## 关闭/删除环境

实验完成之后，请及时关闭或删除Notebook训练作业，避免产生不必要的资源浪费。

### 关闭环境

可点击上图右上角的“终止按钮”关闭环境，亦可在[ModelArts控制台](https://console.huaweicloud.com/modelarts/?region=cn-north-4" \l "/dev-container)的操作栏选择“停止”操作，如下图所示：



关闭环境

### 删除环境

关闭环境后，由于云硬盘从Notebook实例创建成功起直至删除成功，每GB按照规定费用收费，因此当不再使用该云环境时，建议删除环境。

在[ModelArts控制台](https://console.huaweicloud.com/modelarts/?region=cn-north-4" \l "/dev-container)的操作栏打开“更多”，选择“删除”操作，如下图所示：



删除环境

至此ModelArts的人工智能的训练环境搭建完成。

# 基于MindSpore的Bert微调及推理实验

## 模型简介

BERT全称是来自变换器的双向编码器表征量（Bidirectional Encoder Representations from Transformers），它是Google于2018年末开发并发布的一种新型语言模型。与BERT模型相似的预训练语言模型例如问答、命名实体识别、自然语言推理、文本分类等在许多自然语言处理任务中发挥着重要作用。模型是基于Transformer中的Encoder并加上双向的结构，因此一定要熟练掌握Transformer的Encoder的结构。

BERT模型的主要创新点都在pre-train方法上，即用了Masked Language Model和Next Sentence Prediction两种方法分别捕捉词语和句子级别的representation。

在用Masked Language Model方法训练BERT的时候，随机把语料库中15%的单词做Mask操作。对于这15%的单词做Mask操作分为三种情况：80%的单词直接用[Mask]替换、10%的单词直接替换成另一个新的单词、10%的单词保持不变。

因为涉及到Question Answering (QA) 和 Natural Language Inference (NLI)之类的任务，增加了Next Sentence Prediction预训练任务，目的是让模型理解两个句子之间的联系。与Masked Language Model任务相比，Next Sentence Prediction更简单些，训练的输入是句子A和B，B有一半的几率是A的下一句，输入这两个句子，BERT模型预测B是不是A的下一句。

BERT预训练之后，会保存它的Embedding table和12层Transformer权重（BERT-BASE）或24层Transformer权重（BERT-LARGE）。使用预训练好的BERT模型可以对下游任务进行Fine-tuning，比如：文本分类、相似度判断、阅读理解等。

对话情绪识别（Emotion Detection，简称EmoTect），专注于识别智能对话场景中用户的情绪，针对智能对话场景中的用户文本，自动判断该文本的情绪类别并给出相应的置信度，情绪类型分为积极、消极、中性。 对话情绪识别适用于聊天、客服等多个场景，能够帮助企业更好地把握对话质量、改善产品的用户交互体验，也能分析客服服务质量、降低人工质检成本。

## 实验环境准备

新建notebook，运行下面命令，完成MindSpore、MindNLP安装。

安装MindSpore2.2.13

#下载

!wget <https://ms-release.obs.cn-north-4.myhuaweicloud.com/2.2.13/MindSpore/unified/aarch64/mindspore-2.2.13-cp39-cp39-linux_aarch64.whl>

#安装

!pip install mindspore-2.2.13-cp39-cp39-linux\_aarch64.whl --trusted-host ms-release.obs.cn-north-4.myhuaweicloud.com -i <https://pypi.tuna.tsinghua.edu.cn/simple>

安装MindNLP

!pip install mindnlp

## 实验过程

以下代码运行在notebook中

导入必要的库

import os

import mindspore

from mindspore.dataset import text, GeneratorDataset, transforms

from mindspore import nn, context

from mindnlp.\_legacy.engine import Trainer, Evaluator

from mindnlp.\_legacy.engine.callbacks import CheckpointCallback, BestModelCallback

from mindnlp.\_legacy.metrics import Accuracy

定义预数据集

# prepare dataset

class SentimentDataset:

"""Sentiment Dataset"""

def \_\_init\_\_(self, path):

self.path = path

self.\_labels, self.\_text\_a = [], []

self.\_load()

def \_load(self):

with open(self.path, "r", encoding="utf-8") as f:

dataset = f.read()

lines = dataset.split("\n")

for line in lines[1:-1]:

label, text\_a = line.split("\t")

self.\_labels.append(int(label))

self.\_text\_a.append(text\_a)

def \_\_getitem\_\_(self, index):

return self.\_labels[index], self.\_text\_a[index]

def \_\_len\_\_(self):

return len(self.\_labels)

数据集准备

这里提供一份已标注的、经过分词预处理的机器人聊天数据集，来自于百度飞桨团队。数据由两列组成，以制表符（'\t'）分隔，第一列是情绪分类的类别（0表示消极；1表示中性；2表示积极），第二列是以空格分词的中文文本，如下示例，文件为 utf8 编码。

label--text\_a

0--谁骂人了？我从来不骂人，我骂的都不是人，你是人吗 ？

1--我有事等会儿就回来和你聊

2--我见到你很高兴谢谢你帮我

这部分代码主要包括数据集读取，数据格式转换，数据 Tokenize 处理和 pad 操作。

* 数据集下载及解压

# download dataset

!wget https://baidu-nlp.bj.bcebos.com/emotion\_detection-dataset-1.0.0.tar.gz -O emotion\_detection.tar.gz

!tar xvf emotion\_detection.tar.gz

* 数据加载和数据预处理

import numpy as np

def process\_dataset(source, tokenizer, max\_seq\_len=64, batch\_size=32, shuffle=True):

is\_ascend = mindspore.get\_context('device\_target') == 'Ascend'

column\_names = ["label", "text\_a"]

dataset = GeneratorDataset(source, column\_names=column\_names, shuffle=shuffle)

# transforms

type\_cast\_op = transforms.TypeCast(mindspore.int32)

def tokenize\_and\_pad(text):

if is\_ascend:

tokenized = tokenizer(text, padding='max\_length', truncation=True, max\_length=max\_seq\_len)

else:

tokenized = tokenizer(text)

return tokenized['input\_ids'], tokenized['attention\_mask']

# map dataset

dataset = dataset.map(operations=tokenize\_and\_pad, input\_columns="text\_a", output\_columns=['input\_ids', 'attention\_mask'])

dataset = dataset.map(operations=[type\_cast\_op], input\_columns="label", output\_columns='labels')

# batch dataset

if is\_ascend:

dataset = dataset.batch(batch\_size)

else:

dataset = dataset.padded\_batch(batch\_size, pad\_info={'input\_ids': (None, tokenizer.pad\_token\_id),

'attention\_mask': (None, 0)})

return dataset

昇腾NPU环境下暂不支持动态Shape，数据预处理部分采用静态Shape处理

from mindnlp.transformers import BertTokenizer

tokenizer = BertTokenizer.from\_pretrained('bert-base-chinese')

print(tokenizer.pad\_token\_id)

输出：

0

创建训练、验证、测试数据集

dataset\_train = process\_dataset(SentimentDataset("data/train.tsv"), tokenizer)

dataset\_val = process\_dataset(SentimentDataset("data/dev.tsv"), tokenizer)

dataset\_test = process\_dataset(SentimentDataset("data/test.tsv"), tokenizer, shuffle=False)

输出列名

dataset\_train.get\_col\_names()

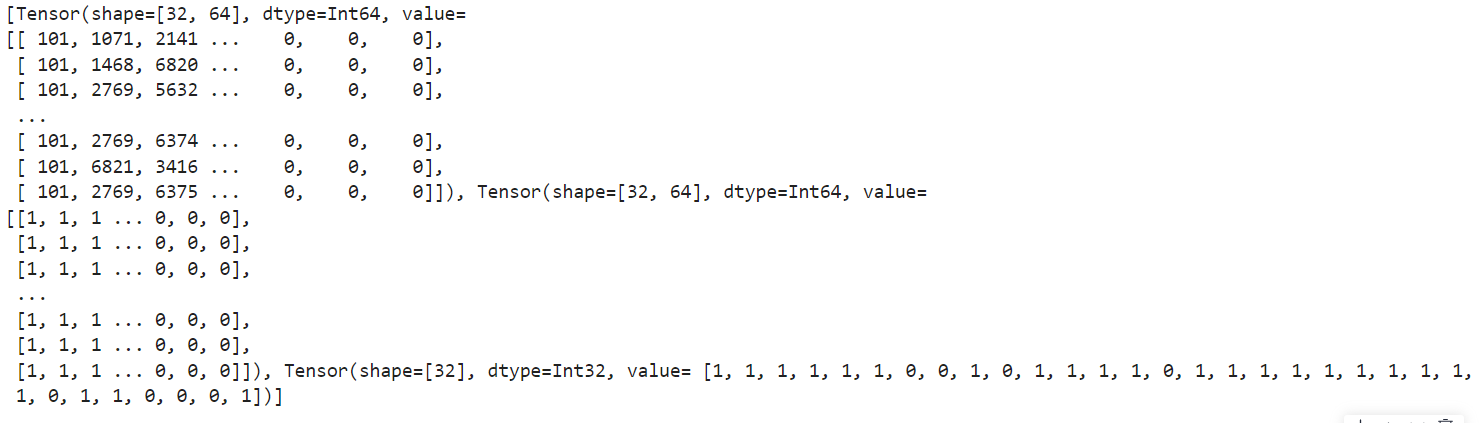
输出：

['input\_ids', 'attention\_mask', 'labels']

数据集取出测试

print(next(dataset\_train.create\_tuple\_iterator()))

输出：



模型构建

通过 BertForSequenceClassification 构建用于情感分类的 BERT 模型，加载预训练权重，设置情感三分类的超参数自动构建模型。后面对模型采用自动混合精度操作，提高训练的速度，然后实例化优化器，紧接着实例化评价指标，设置模型训练的权重保存策略，最后就是构建训练器，模型开始训练。

* 实例优化器

from mindnlp.transformers import BertForSequenceClassification, BertModel

from mindnlp.\_legacy.amp import auto\_mixed\_precision

# set bert config and define parameters for training

model = BertForSequenceClassification.from\_pretrained('bert-base-chinese', num\_labels=3)

model = auto\_mixed\_precision(model, 'O1')

optimizer = nn.Adam(model.trainable\_params(), learning\_rate=2e-5)

* 实例化评估指标

metric = Accuracy()

# define callbacks to save checkpoints

ckpoint\_cb = CheckpointCallback(save\_path='checkpoint', ckpt\_name='bert\_emotect', epochs=1, keep\_checkpoint\_max=2)

best\_model\_cb = BestModelCallback(save\_path='checkpoint', ckpt\_name='bert\_emotect\_best', auto\_load=True)

trainer = Trainer(network=model, train\_dataset=dataset\_train,

eval\_dataset=dataset\_val, metrics=metric,

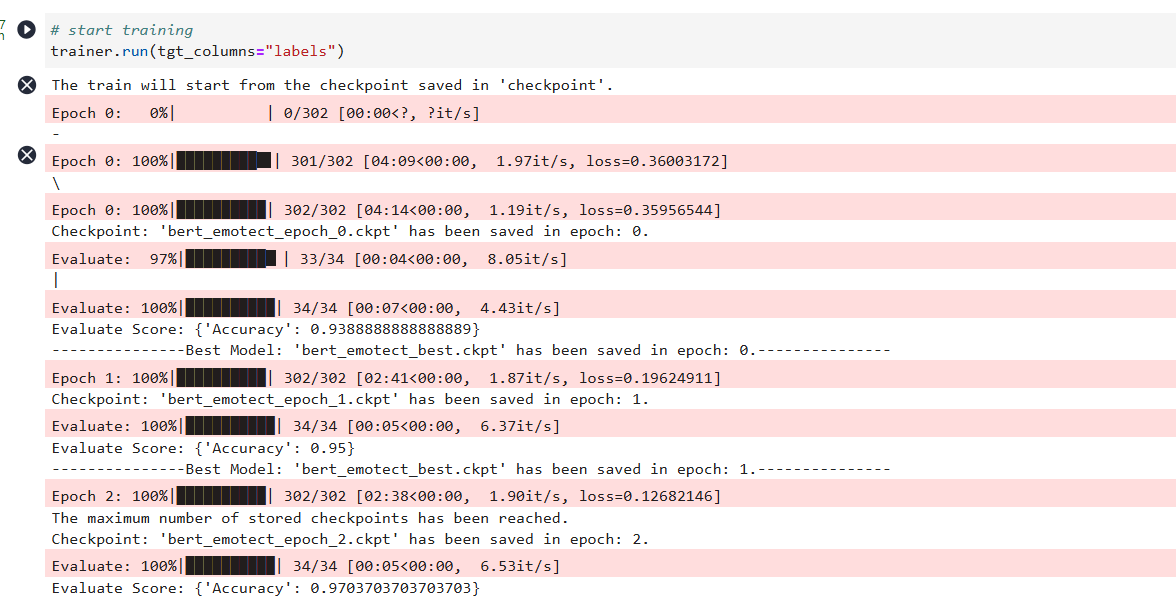
epochs=5, optimizer=optimizer, callbacks=[ckpoint\_cb, best\_model\_cb])

启动训练

# start training

trainer.run(tgt\_columns="labels")

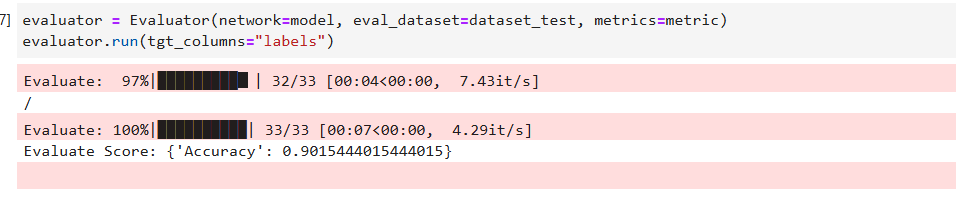
训练结果：



精度校验结果

evaluator = Evaluator(network=model, eval\_dataset=dataset\_test, metrics=metric)

evaluator.run(tgt\_columns="labels")



## 模型推理

遍历推理数据集，将结果与标签进行统一展示。

dataset\_infer = SentimentDataset("data/infer.tsv")

def predict(text, label=None):

label\_map = {0: "消极", 1: "中性", 2: "积极"}

text\_tokenized = Tensor([tokenizer(text).input\_ids])

logits = model(text\_tokenized)

predict\_label = logits[0].asnumpy().argmax()

info = f"inputs: '{text}', predict: '{label\_map[predict\_label]}'"

if label is not None:

info += f" , label: '{label\_map[label]}'"

print(info)

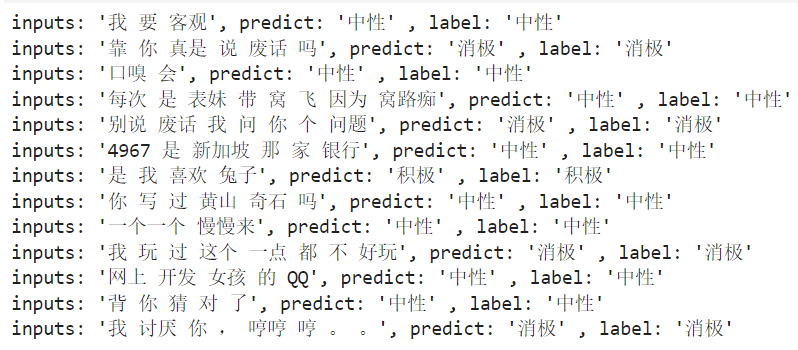
* 推理数据集展示

from mindspore import Tensor

for label, text in dataset\_infer:

predict(text, label)

输出：



自己输入推理数据，展示模型的泛化能力

predict("家人们咱就是说一整个无语住了 绝绝子叠buff")

输出：

