《自然语言处理》

-文本分类



华为技术有限公司

|  |
| --- |
| 版权所有 © 华为技术有限公司 2022。 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  C:\Users\jwx341670\Desktop\华为标志 Huawei Logo 2018\竖版标志Vertical Version\PNG\HW_POS_RBG_Vertical-150ppi.png 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 实验总览 2](#_Toc151383334)

[1.1 实验背景 2](#_Toc151383335)

[1.2 实验目的 2](#_Toc151383336)

[1.3 实验清单 2](#_Toc151383337)

[2 TextCNN情感实验 3](#_Toc151383338)

[2.1 实验简介 3](#_Toc151383339)

[2.2 实验环境 3](#_Toc151383340)

[2.3 实验步骤 3](#_Toc151383341)

[2.3.1 实验过程 3](#_Toc151383342)

[2.4 实验小结 14](#_Toc151383343)

# 实验总览

## 实验背景

文本分类( text classification),又称文档分类( document classification),指的是将一个文档归类到一个或多个类别中的自然语言处理任务。文本分类的应用场景非常广泛，涵盖垃圾邮件过滤、垃圾评论过滤、自动标签、情感分析等任何需要自动归档文本的场合。

## 实验目的

* 理解文本分类的基本流程
* 理解CNN网络在文本任务中的用法
* 掌握MindSpore搭建文本分类模型的方法

## 实验清单

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 实验 | 简述 | 难度 | 软件环境 | 开发环境 |
| TextCNN情感分析实验 | 基于MindSpore搭建TextCNN模型用于情感分析 | 中级 | MindSpore 2.1 | ModelArts Ascend Notebook环境 |

# TextCNN情感实验

## 实验简介

情感分析是自然语言处理文本分类任务的应用场景之一，情感分类较为简单，实用性也较强。常见的购物网站、电影网站都可以采集到相对高质量的数据集，也很容易给业务领域带来收益。例如，可以结合领域上下文，自动分析特定类型客户对当前产品的意见，可以分主题分用户类型对情感进行分析，以作针对性的处理，甚至基于此进一步推荐产品，提高转化率，带来更高的商业收益。

本实验主要基于卷积神经网络对电影评论信息进行情感分析，判断其情感倾向。

## 实验环境

在开始本实验前，需要完成实验环境搭建工作。

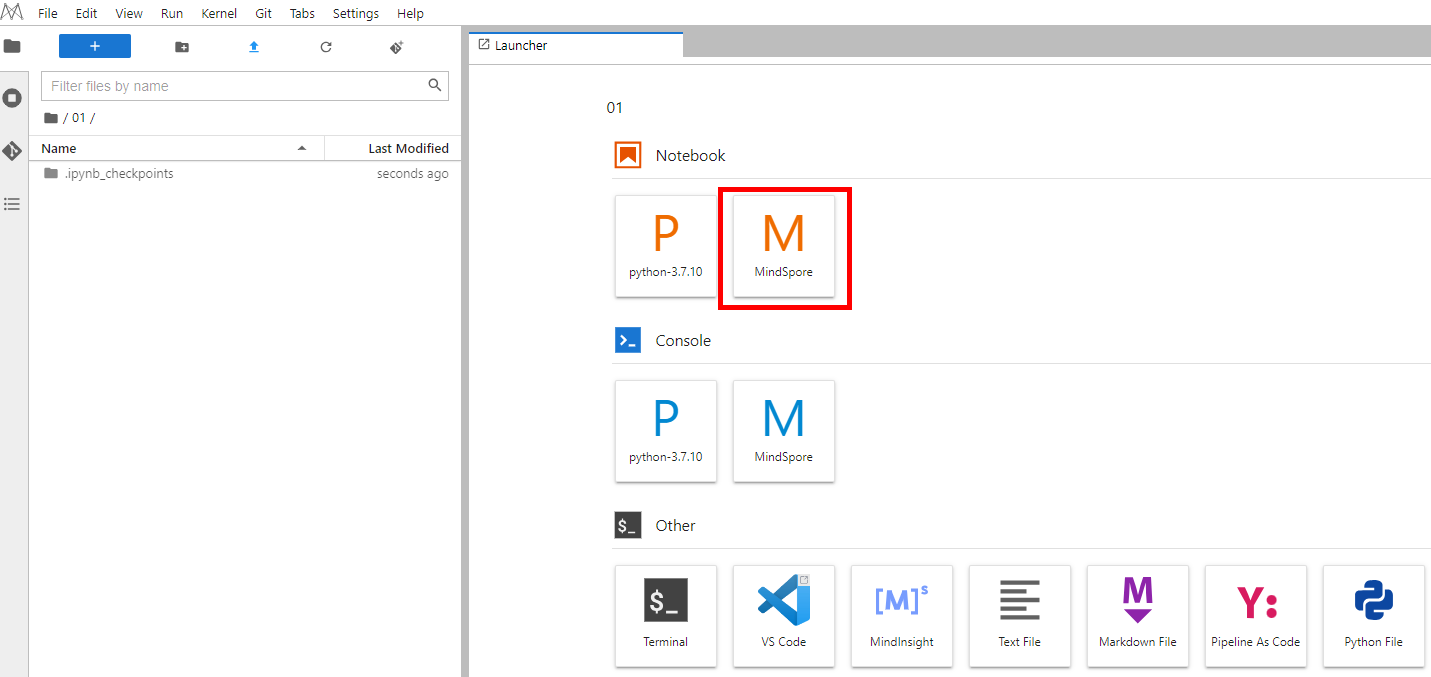
进入ModelArts开发环境

环境搭建方式可参考下方环境搭建手册中 “2.华为云ModelArts环境搭建（训练环境）”部分内容。



打开Notebook

打开Notebook控制台后，新建或打开ipynb文件，选择MindSpore环境作为Kernel，即可开始编辑实验代码。



创建ipynb文件

下载数据集

在打开的Notebook中输入以下命令，将本实验所需的数据集下载到Notebook环境并解压。

#设置no\_proxy环境变量

%env no\_proxy='a.test.com,127.0.0.1,2.2.2.2'

!wget <https://ascend-professional-construction-dataset.obs.cn-north-4.myhuaweicloud.com:443/NLP/data0.zip>

!unzip data0.zip

## 实验步骤

### 实验过程

导入依赖库

输入：

import math

import numpy as np

import pandas as pd

import os

import math

import random

import codecs

from pathlib import Path

import mindspore

import mindspore.dataset as ds

import mindspore.nn as nn

from mindspore import Tensor

from mindspore import context

from mindspore.train.model import Model

from mindspore.nn.metrics import Accuracy

from mindspore.train.serialization import load\_checkpoint, load\_param\_into\_net

from mindspore.train.callback import ModelCheckpoint, CheckpointConfig, LossMonitor, TimeMonitor

from mindspore.ops import operations as ops

超参数设置

输入：

from easydict import EasyDict as edict

cfg = edict({

'name': 'movie review',

'pre\_trained': False,

'num\_classes': 2,

'batch\_size': 64,

'epoch\_size': 4,

'weight\_decay': 3e-5,

'data\_path': './data/',

'device\_target': 'Ascend',

'device\_id': 0,

'keep\_checkpoint\_max': 1,

'checkpoint\_path': './ckpt/train\_textcnn-4\_149.ckpt',

'word\_len': 51,

'vec\_length': 40

})

运行环境设置

输入：

context.set\_context(mode=context.GRAPH\_MODE, device\_target=cfg.device\_target, device\_id=cfg.device\_id)

数据预览

输入：

# 数据预览

with open("./data/rt-polarity.neg", 'r', encoding='utf-8') as f:

print("Negative reivews:")

for i in range(5):

print("[{0}]:{1}".format(i,f.readline()))

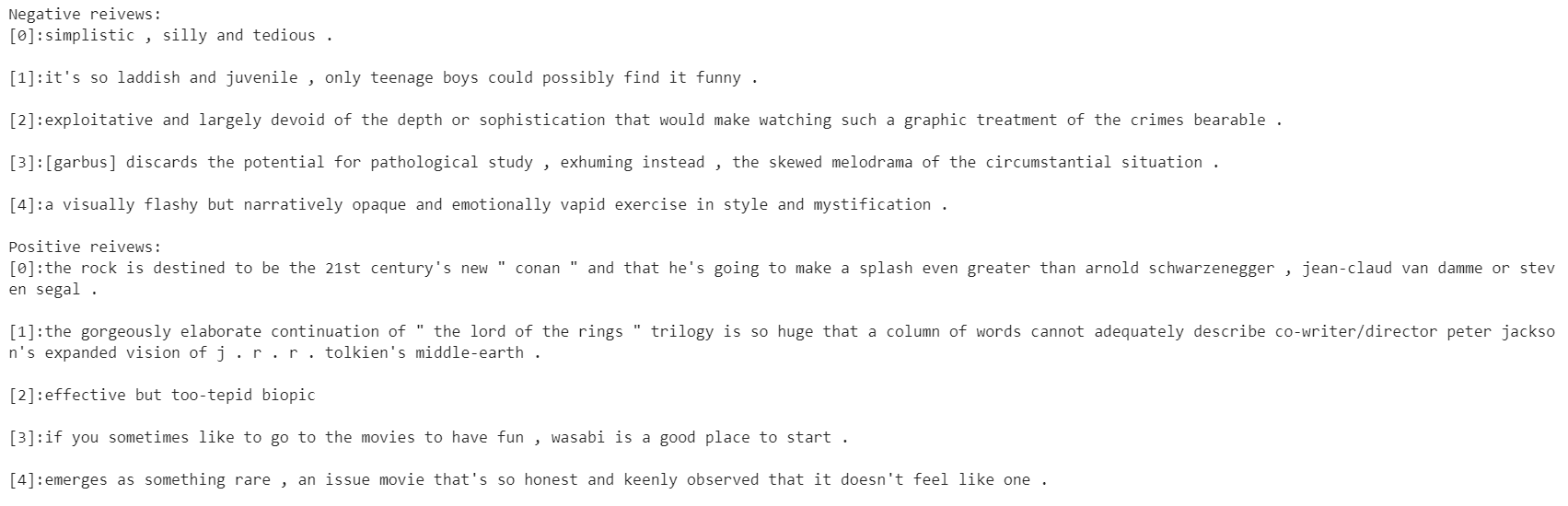
with open("./data/rt-polarity.pos", 'r', encoding='utf-8') as f:

print("Positive reivews:")

for i in range(5):

print("[{0}]:{1}".format(i,f.readline()))

输出：



文本数据内容展示

定义数据预处理函数

输入：

#定义数据生成类

class Generator():

def \_\_init\_\_(self, input\_list):

self.input\_list=input\_list

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

return (np.array(self.input\_list[item][0],dtype=np.int32),

np.array(self.input\_list[item][1],dtype=np.int32))

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

return len(self.input\_list)

输入

class MovieReview:

'''

影评数据集

'''

def \_\_init\_\_(self, root\_dir, maxlen, split):

'''

input:

root\_dir: 影评数据目录

maxlen: 设置句子最大长度

split: 设置数据集中训练/评估的比例

'''

self.path = root\_dir

self.feelMap = {

'neg':0,

'pos':1

}

self.files = []

self.doConvert = False

mypath = Path(self.path)

if not mypath.exists() or not mypath.is\_dir():

print("please check the root\_dir!")

raise ValueError

# 在数据目录中找到文件

for root,\_,filename in os.walk(self.path):

for each in filename:

self.files.append(os.path.join(root,each))

break

# 确认是否为两个文件.neg与.pos

if len(self.files) != 2:

print("There are {} files in the root\_dir".format(len(self.files)))

raise ValueError

# 读取数据

self.word\_num = 0

self.maxlen = 0

self.minlen = float("inf")

self.maxlen = float("-inf")

self.Pos = []

self.Neg = []

for filename in self.files:

self.read\_data(filename)

self.text2vec(maxlen=maxlen)

self.split\_dataset(split=split)

def read\_data(self, filePath):

with open(filePath,'r') as f:

for sentence in f.readlines():

sentence = sentence.replace('\n','')\

.replace('"','')\

.replace('\'','')\

.replace('.','')\

.replace(',','')\

.replace('[','')\

.replace(']','')\

.replace('(','')\

.replace(')','')\

.replace(':','')\

.replace('--','')\

.replace('-',' ')\

.replace('\\','')\

.replace('0','')\

.replace('1','')\

.replace('2','')\

.replace('3','')\

.replace('4','')\

.replace('5','')\

.replace('6','')\

.replace('7','')\

.replace('8','')\

.replace('9','')\

.replace('`','')\

.replace('=','')\

.replace('$','')\

.replace('/','')\

.replace('\*','')\

.replace(';','')\

.replace('<b>','')\

.replace('%','')

sentence = sentence.split(' ')

sentence = list(filter(lambda x: x, sentence))

if sentence:

self.word\_num += len(sentence)

self.maxlen = self.maxlen if self.maxlen >= len(sentence) else len(sentence)

self.minlen = self.minlen if self.minlen <= len(sentence) else len(sentence)

if 'pos' in filePath:

self.Pos.append([sentence,self.feelMap['pos']])

else:

self.Neg.append([sentence,self.feelMap['neg']])

def text2vec(self, maxlen):

'''

将句子转化为向量

'''

# Vocab = {word : index}

self.Vocab = dict()

# self.Vocab['None']

for SentenceLabel in self.Pos+self.Neg:

vector = [0]\*maxlen

for index, word in enumerate(SentenceLabel[0]):

if index >= maxlen:

break

if word not in self.Vocab.keys():

self.Vocab[word] = len(self.Vocab)

vector[index] = len(self.Vocab) - 1

else:

vector[index] = self.Vocab[word]

SentenceLabel[0] = vector

self.doConvert = True

def split\_dataset(self, split):

'''

分割为训练集与测试集

'''

trunk\_pos\_size = math.ceil((1-split)\*len(self.Pos))

trunk\_neg\_size = math.ceil((1-split)\*len(self.Neg))

trunk\_num = int(1/(1-split))

pos\_temp=list()

neg\_temp=list()

for index in range(trunk\_num):

pos\_temp.append(self.Pos[index\*trunk\_pos\_size:(index+1)\*trunk\_pos\_size])

neg\_temp.append(self.Neg[index\*trunk\_neg\_size:(index+1)\*trunk\_neg\_size])

self.test = pos\_temp.pop(2)+neg\_temp.pop(2)

self.train = [i for item in pos\_temp+neg\_temp for i in item]

random.shuffle(self.train)

# random.shuffle(self.test)

def get\_dict\_len(self):

'''

获得数据集中文字组成的词典长度

'''

if self.doConvert:

return len(self.Vocab)

else:

print("Haven't finished Text2Vec")

return -1

def create\_train\_dataset(self, epoch\_size, batch\_size):

dataset = ds.GeneratorDataset(

source=Generator(input\_list=self.train),

column\_names=["data","label"],

shuffle=False

)

dataset=dataset.batch(batch\_size=batch\_size,drop\_remainder=True)

dataset=dataset.repeat(epoch\_size)

return dataset

def create\_test\_dataset(self, batch\_size):

dataset = ds.GeneratorDataset(

source=Generator(input\_list=self.test),

column\_names=["data","label"],

shuffle=False

)

dataset=dataset.batch(batch\_size=batch\_size,drop\_remainder=True)

return dataset

生成数据集

生成数据类实例，获得训练用数据集

输入：

instance = MovieReview(root\_dir=cfg.data\_path, maxlen=cfg.word\_len, split=0.9)

dataset = instance.create\_train\_dataset(batch\_size=cfg.batch\_size,epoch\_size=cfg.epoch\_size)

batch\_num = dataset.get\_dataset\_size()

展示结果：

vocab\_size=instance.get\_dict\_len()

print("vocab\_size:{0}".format(vocab\_size))

item =dataset.create\_dict\_iterator()

for i,data in enumerate(item):

if i<1:

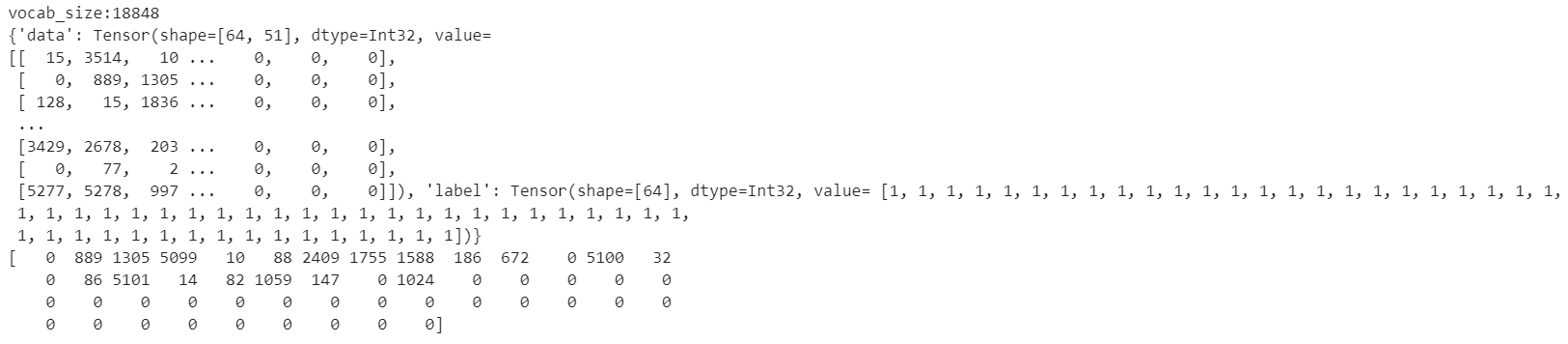
print(data)

print(data['data'][1])

else:

break

输出：



预处理结果展示

训练参数设置

学习率设置

输入：

learning\_rate = []

warm\_up = [1e-3 / math.floor(cfg.epoch\_size / 5) \* (i + 1) for \_ in range(batch\_num)

for i in range(math.floor(cfg.epoch\_size / 5))]

shrink = [1e-3 / (16 \* (i + 1)) for \_ in range(batch\_num)

for i in range(math.floor(cfg.epoch\_size \* 3 / 5))]

normal\_run = [1e-3 for \_ in range(batch\_num) for i in

range(cfg.epoch\_size - math.floor(cfg.epoch\_size / 5)

- math.floor(cfg.epoch\_size \* 2 / 5))]

learning\_rate = learning\_rate + warm\_up + normal\_run + shrink

定义TextCNN

模型类定义了模型结构搭建、训练、评估、加载离线模型、在线推理函数。

输入：

class TextCNN(nn.Cell):

def \_\_init\_\_(self, vocab\_len, word\_len, num\_classes, vec\_length):

super(TextCNN, self).\_\_init\_\_()

self.vec\_length = vec\_length

self.word\_len = word\_len

self.num\_classes = num\_classes

self.embedding = nn.Embedding(vocab\_len, self.vec\_length, embedding\_table='normal')

# 请同学们补全init方法，以定义CNN模型结构

# 代码补全结束

def construct(self,x):

# 请同学们补全construct方法，以构建CNN计算逻辑

# 代码补全结束

return x

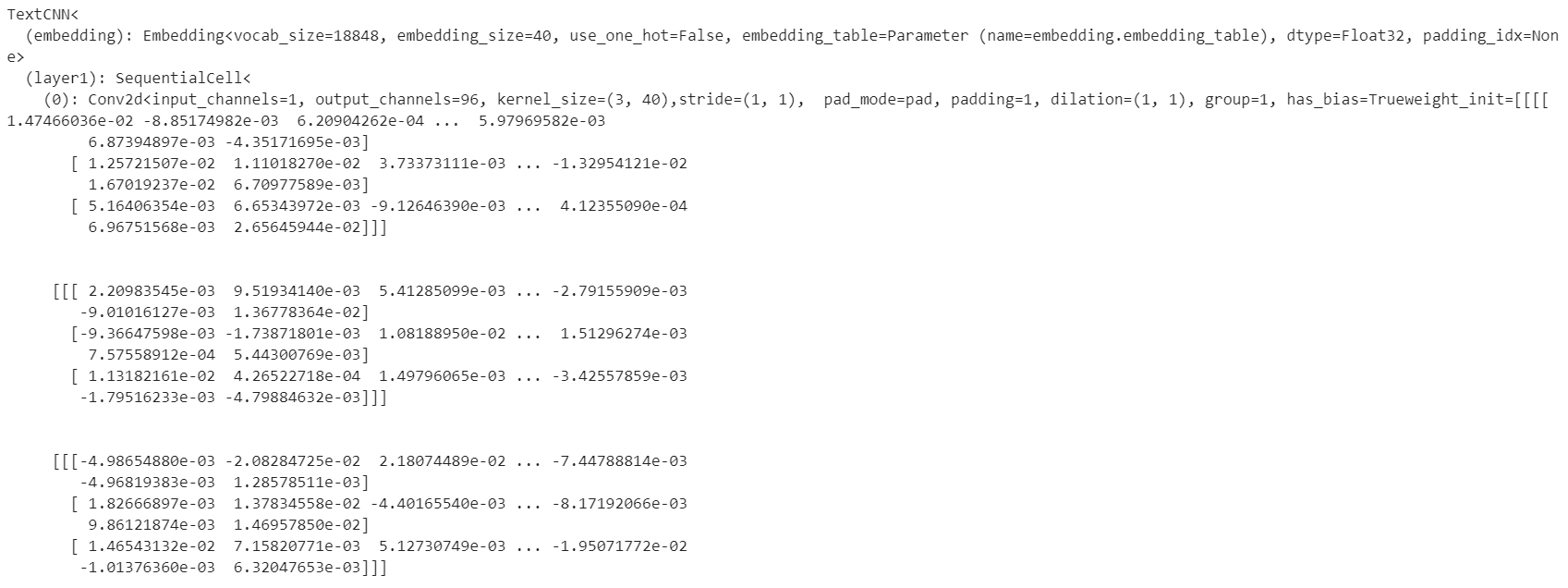
实例化

net = TextCNN(vocab\_len=instance.get\_dict\_len(), word\_len=cfg.word\_len,

num\_classes=cfg.num\_classes, vec\_length=cfg.vec\_length)

print(net)

查看神经网络概况



神经网络结构展示

定义训练相关参数

# 优化器、损失函数、保存检查点、时间监视器等设置

opt = nn.Adam(filter(lambda x: x.requires\_grad, net.get\_parameters()),

learning\_rate=learning\_rate, weight\_decay=cfg.weight\_decay)

loss = nn.SoftmaxCrossEntropyWithLogits(sparse=True)

model = Model(net, loss\_fn=loss, optimizer=opt, metrics={'acc': Accuracy()})

config\_ck = CheckpointConfig(save\_checkpoint\_steps=int(cfg.epoch\_size\*batch\_num/2), keep\_checkpoint\_max=cfg.keep\_checkpoint\_max)

time\_cb = TimeMonitor(data\_size=batch\_num)

ckpt\_save\_dir = "./ckpt"

ckpoint\_cb = ModelCheckpoint(prefix="train\_textcnn", directory=ckpt\_save\_dir, config=config\_ck)

loss\_cb = LossMonitor()

启动训练

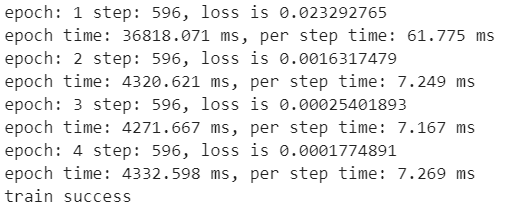
开展训练

输入：

model.train(cfg.epoch\_size, dataset, callbacks=[time\_cb, ckpoint\_cb, loss\_cb])

print("train success")

输出：



训练过程展示

测试评估

输入：

# 导入训练生成的checkpoint

checkpoint\_path = './ckpt/train\_textcnn-4\_596.ckpt'

# 生成测试数据集

dataset = instance.create\_test\_dataset(batch\_size=cfg.batch\_size)

opt = nn.Adam(filter(lambda x: x.requires\_grad, net.get\_parameters()),

learning\_rate=0.001, weight\_decay=cfg.weight\_decay)

loss = nn.SoftmaxCrossEntropyWithLogits(sparse=True)

net = TextCNN(vocab\_len=instance.get\_dict\_len(),word\_len=cfg.word\_len,

num\_classes=cfg.num\_classes,vec\_length=cfg.vec\_length)

if checkpoint\_path is not None:

param\_dict = load\_checkpoint(checkpoint\_path)

print("load checkpoint from [{}].".format(checkpoint\_path))

else:

param\_dict = load\_checkpoint(cfg.checkpoint\_path)

print("load checkpoint from [{}].".format(cfg.checkpoint\_path))

load\_param\_into\_net(net, param\_dict)

net.set\_train(False)

model = Model(net, loss\_fn=loss, metrics={'acc': Accuracy()})

acc = model.eval(dataset)

print("accuracy: ", acc)

输出：



准确率展示

在线测试

定义前处理函数

输入：

def preprocess(sentence):

sentence = sentence.lower().strip()

sentence = sentence.replace('\n','')\

.replace('"','')\

.replace('\'','')\

.replace('.','')\

.replace(',','')\

.replace('[','')\

.replace(']','')\

.replace('(','')\

.replace(')','')\

.replace(':','')\

.replace('--','')\

.replace('-',' ')\

.replace('\\','')\

.replace('0','')\

.replace('1','')\

.replace('2','')\

.replace('3','')\

.replace('4','')\

.replace('5','')\

.replace('6','')\

.replace('7','')\

.replace('8','')\

.replace('9','')\

.replace('`','')\

.replace('=','')\

.replace('$','')\

.replace('/','')\

.replace('\*','')\

.replace(';','')\

.replace('<b>','')\

.replace('%','')\

.replace(" "," ")

sentence = sentence.split(' ')

maxlen = cfg.word\_len

vector = [0]\*maxlen

for index, word in enumerate(sentence):

if index >= maxlen:

break

if word not in instance.Vocab.keys():

print(word,"单词未出现在字典中")

else:

vector[index] = instance.Vocab[word]

sentence = vector

return sentence

def inference(review\_en):

review\_en = preprocess(review\_en)

input\_en = Tensor(np.array([review\_en]).astype(np.int32))

output = net(input\_en)

if np.argmax(np.array(output[0])) == 1:

print("Positive comments")

else:

print("Negative comments")

取单条评论文本数据，进行测试，输出情感类别及其概率

负面样例

输入：

review\_en = "the movie is so boring"

inference(review\_en)

输出：



评论预测结果展示

## 实验小结

本实验介绍了如何使用MindSpore搭建用于文本分类的TextCNN模型，通过实验，使学员了解文本分类任务的基本流程，同时理解卷积网络在文本任务中的使用方法，通过实验也加深了对CNN网络的理解，同时提升了代码实践能力。