《自然语言处理》

-文本分类

华为技术有限公司

目录

1 实验总览

1.1 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。

1.2 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

1.3 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 1.1	ModelArts Ascend Notebook环境

2 TextCNN 情感实验

2.1 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。

2.2 实验环境

ModelArts Ascend Notebook 环境，环境设置参考《华为云 ModelArts Ascend 环境配置》

2.3 实验步骤

2.3.1 实验准备

步骤 1　OBS 创建项目文件夹

使用 OBS Brower+登录 OBS

图2-1 OBS Brower+登录

进入之前创建的用于挂载 notebook 的 obs 目录，此处我们是建了一个“nlp”的目录用于挂载昇腾环境下的 notebook：

图2-2 进入课程主目录

创建“text_classification_mindspore”文件夹

图2-3 创建项目文件夹

创建完如下所示：

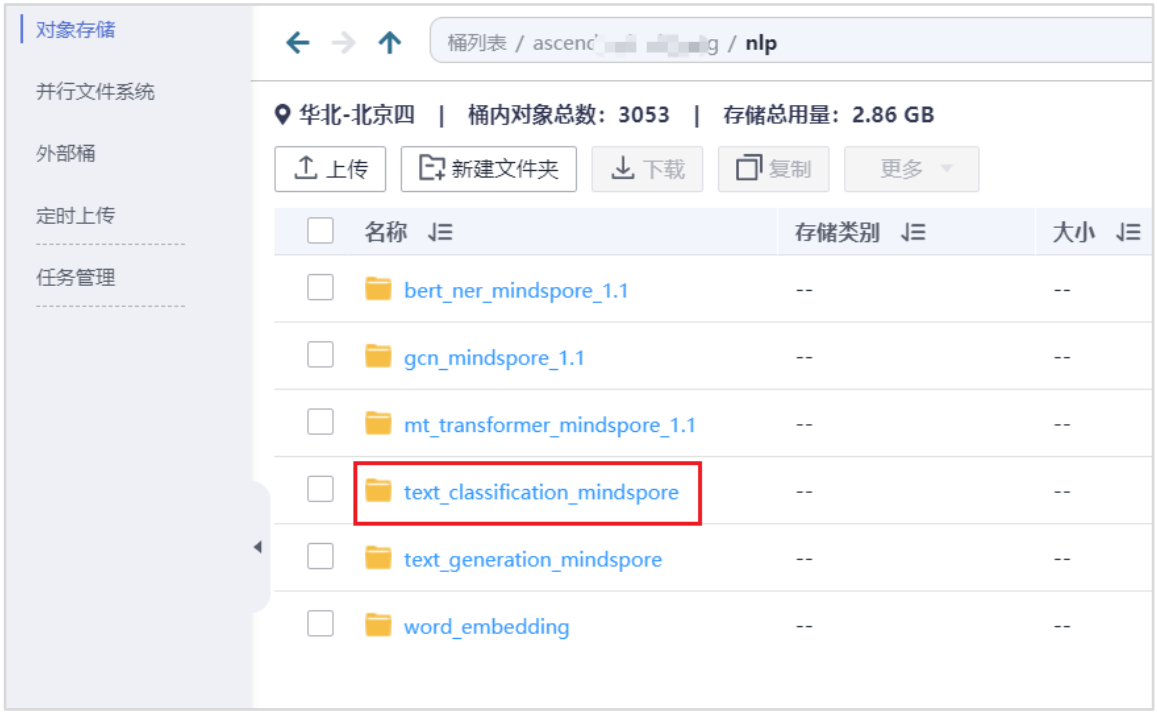

图2-4 项目文件夹创建成功

步骤 2　上传实验源码及数据

进入刚创建的“text_classification_mindspore”文件夹，上传源码及数据至该目录下。

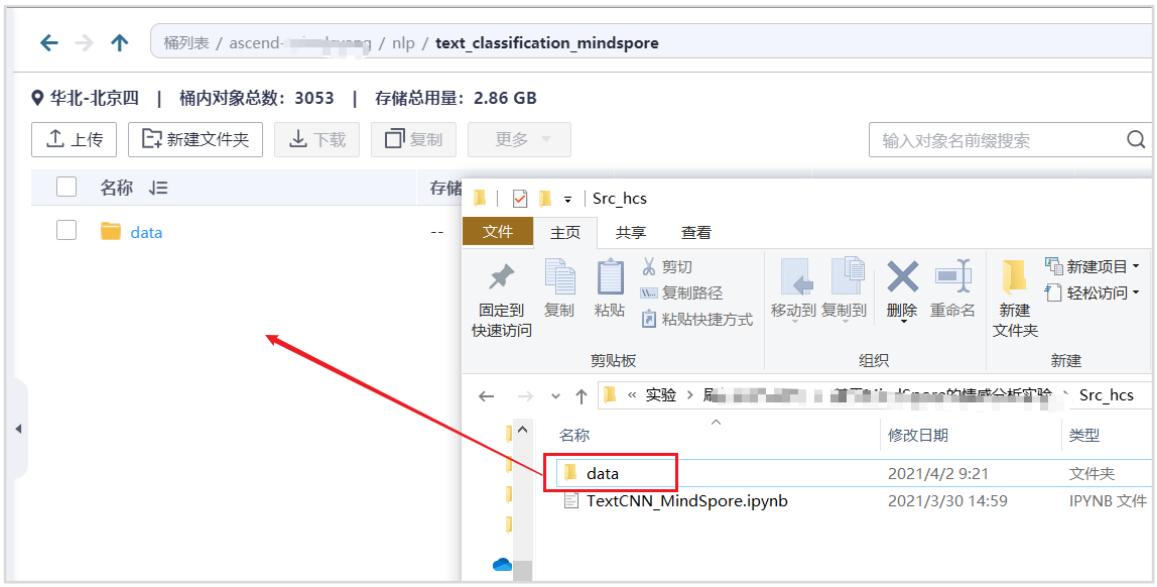

图2-5 上传实验数据

步骤 3　打开 JupyterLab

登录华为云，启动之前创建的 ModelArts Ascend notebook 环境

图2-6 启动 notebook

打开 JupyterLab，并进入之前创建的“text_classification_mindspore”文件夹

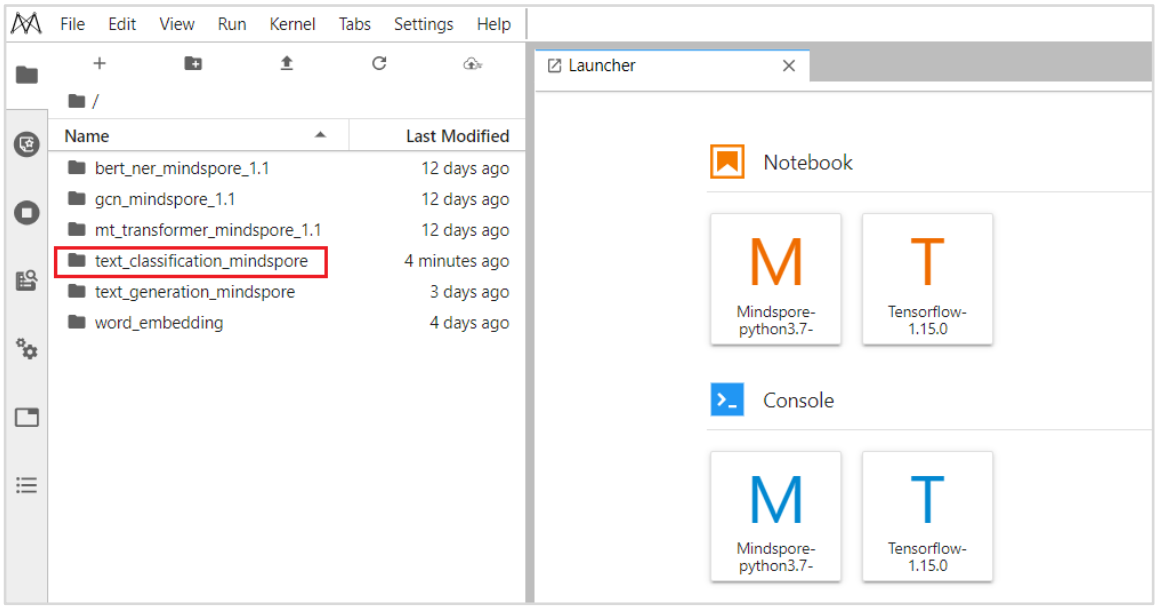

图2-7 进入项目文件夹

新建 notebook 文件：

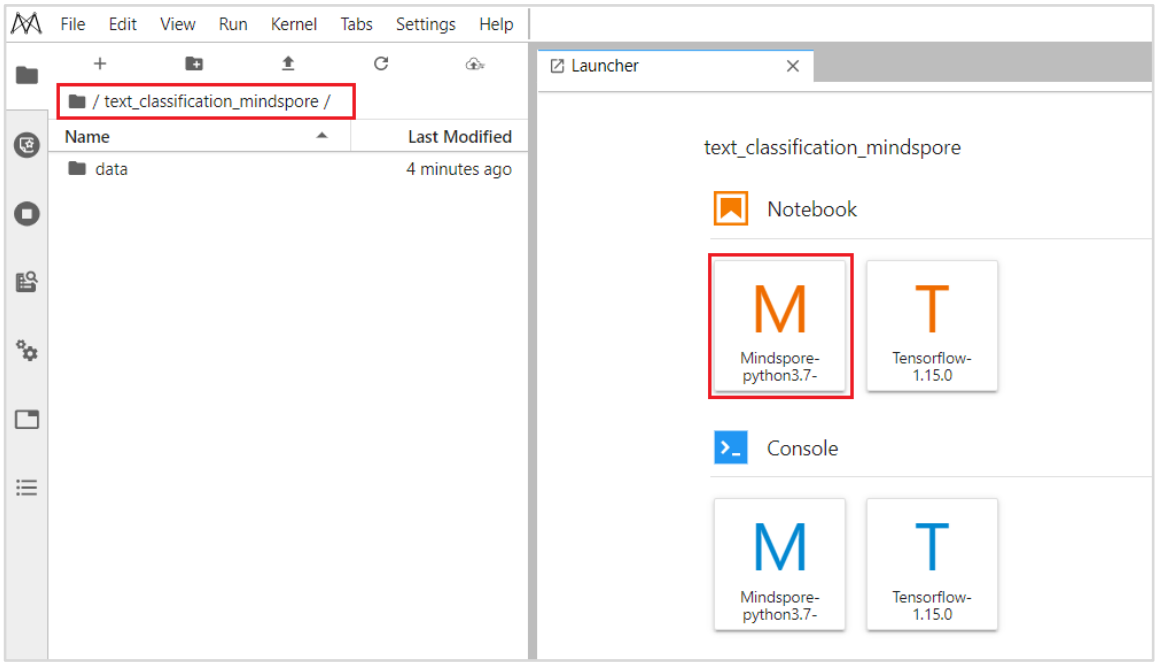

图2-8 新建 notebook 文件

重命名为“TextCNN_MindSpore.ipynb”并打开，右侧显示代码编辑区

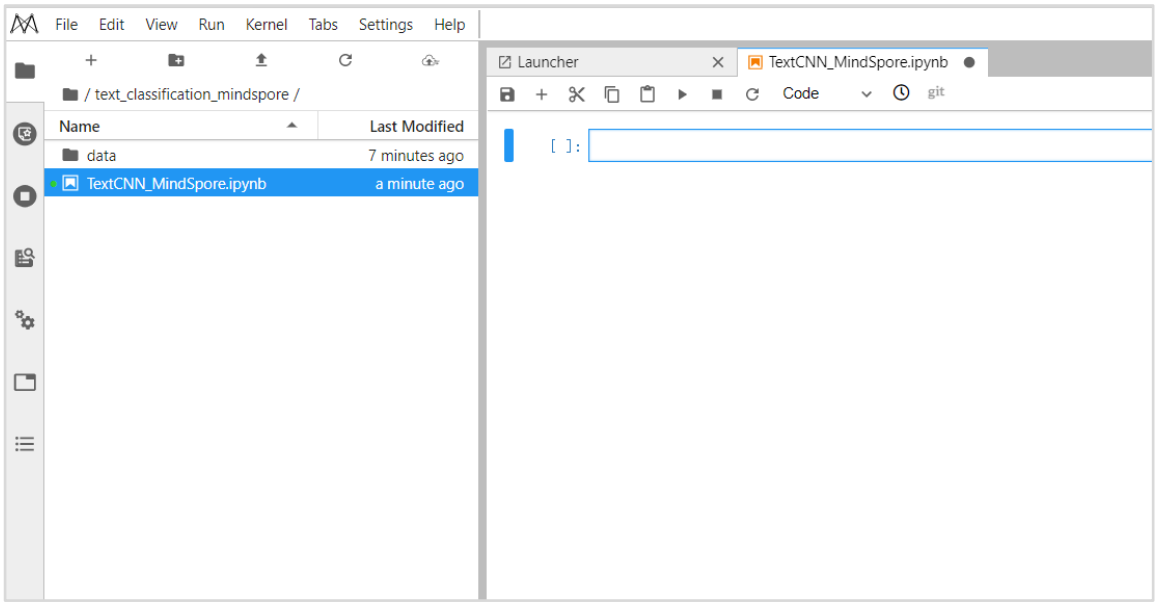

图2-9 重命名并打开 notebook 文件

2.3.2 实验过程

步骤 1　同步数据和源码至本地容器

因为 notebook 是挂载在 obs 上，运行的容器实例不能直接读取操作 obs 上的文件，需下载至容器本地环境中，请在 src_url 字段将 obs 桶名称替换成自己创建的 obs 桶名称。

输入：

```
import moxing as mox
mox.file.copy_parallel(src_url="s3://替换你自己的 obs 路径
/nlp/text_classification_mindspore/data/", dst_url='./data/')
```

步骤 2　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
```

步骤 3　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
})
```

步骤 4　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)
```

步骤 5　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

```
Negative reivews:
[0]:simplistic , silly and tedious .

[1]:it's so laddish and juvenile , only teenage boys could possibly find it funny .

[2]:exploitative and largely devoid of the depth or sophistication that would make watching such a graphic treatment of the crimes bearable .

[3]:[garbus] discards the potential for pathological study , exhuming instead , the skewed melodrama of the circumstantial situation .

[4]:a visually flashy but narratively opaque and emotionally vapid exercise in style and mystification .

Positive reivews:
[0]:the rock is destined to be the 21st century's new " conan " and that he's going to make a splash even greater than arnold schwarzenegger , jean-claud van damme or stev
en segal .

[1]:the gorgeously elaborate continuation of " the lord of the rings " trilogy is so huge that a column of words cannot adequately describe co-writer/director peter jackso
n's expanded vision of j . r . r . tolkien's middle-earth .

[2]:effective but too-tepid biopic

[3]:if you sometimes like to go to the movies to have fun , wasabi is a good place to start .

[4]:emerges as something rare , an issue movie that's so honest and keenly observed that it doesn't feel like one .
```

图2-10 文本数据内容展示

步骤 6　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:
    f = codecs.open(filename, 'r')
    ff = f.read()
    file_object = codecs.open(filename, 'w', 'utf-8')
    file_object.write(ff)
    self.read_data(filename)
self.PosNeg = self.Pos + self.Neg

self.text2vec(maxlen=maxlen)
self.split_dataset(split=split)
```

```
def read_data(self, filePath):
# 数据预处理，替代特殊符号，将句子分割为单词
    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:
            ##########请将代码补充完整##############
        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):
        # 生成训练能用数据集
        dataset = ds.GeneratorDataset(
                                            source=Generator(input_list=self.train),
                                            column_names=["data","label"],
                                            shuffle=False
```

```
                                              )
#               dataset.set_dataset_size(len(self.train))
            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.set_dataset_size(len(self.test))
            dataset=dataset.batch(batch_size=batch_size,drop_remainder=True)
            return dataset
```

步骤 7　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<2:
            print(data)
            print(data['data'][1])
      else:
            break
```

输出：

```
vocab_size:18848
{'data': Tensor(shape=[64, 51], dtype=Int32, value=
[[  15, 3514,   10 ...    0,    0,    0],
 [   0,  889, 1305 ...    0,    0,    0],
 [ 128,   15, 1836 ...    0,    0,    0],
 ...
 [3429, 2678,  203 ...    0,    0,    0],
 [   0,   77,    2 ...    0,    0,    0],
 [5277, 5278,  997 ...    0,    0,    0]]), 'label': Tensor(shape=[64], dtype=Int32, value= [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1])}
[   0  889 1305 5099   10   88 2409 1755 1588  186  672    0 5100   32
    0   86 5101   14   82 1059  147    0 1024    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0]
```

步骤 8　训练参数设置

学习率设置

输入：

```
# 思考题，这里的 warm_up，normal_run，shrink 几个阶段比例如何
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
```

步骤 9　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：

```
def _weight_variable(shape, factor=0.01):
    init_value = np.random.randn(*shape).astype(np.float32) * factor
    return Tensor(init_value)


def make_conv_layer(kernel_size):
    weight_shape = (96, 1, *kernel_size)
    weight = _weight_variable(weight_shape)
    return nn.Conv2d(in_channels=1, out_channels=96, kernel_size=kernel_size, padding=1,
                     pad_mode="pad", weight_init=weight, has_bias=True)


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.unsqueeze = ops.ExpandDims()
        self.embedding = nn.Embedding(vocab_len, self.vec_length, embedding_table='normal')

        self.slice = ops.Slice()
        self.layer1 = self.make_layer(kernel_height=3)
        self.layer2 = self.make_layer(kernel_height=4)
        self.layer3 = self.make_layer(kernel_height=5)
```

```
        self.concat = ops.Concat(1)

        self.fc = nn.Dense(96*3, self.num_classes)
        self.drop = nn.Dropout(keep_prob=0.5)
        self.print = ops.Print()
        self.reducemean = ops.ReduceMax(keep_dims=False)

    def make_layer(self, kernel_height):
        return nn.SequentialCell(
            [
                make_conv_layer((kernel_height,self.vec_length)),
                nn.ReLU(),
                nn.MaxPool2d(kernel_size=(self.word_len-kernel_height+1,1)),
            ]
        )

    def construct(self,x):
        x = self.unsqueeze(x, 1)
        x = self.embedding(x)
        x1 = self.layer1(x)
        x2 = self.layer2(x)
        x3 = self.layer3(x)

        x1 = self.reducemean(x1, (2, 3))
        x2 = self.reducemean(x2, (2, 3))
        x3 = self.reducemean(x3, (2, 3))

        x = self.concat((x1, x2, x3))
        x = self.drop(x)
        x = self.fc(x)
        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)
```

查看神经网络概况

```
TextCNN<
  (embedding): Embedding<vocab_size=18848, embedding_size=40, use_one_hot=False, embedding_table=Parameter (name=embedding.embedding_table), dtype=Float32, padding_idx=None>
  (layer1): SequentialCell<
    (0): Conv2d<input_channels=1, output_channels=96, kernel_size=(3, 40),stride=(1, 1),  pad_mode=pad, padding=1, dilation=(1, 1), group=1, has_bias=Trueweight_init=[[[[ 1.47466036e-02 -8.85174982e-03  6.20904262e-04 ...  5.97969582e-03
          6.87394897e-03 -4.35171695e-03]
        [ 1.25721507e-02  1.11018270e-02  3.73373111e-03 ... -1.32954121e-02
          1.67019237e-02  6.70977589e-03]
        [ 5.16406354e-03  6.65343972e-03 -9.12646390e-03 ...  4.12355090e-04
          6.96751568e-03  2.65645944e-02]]]


     [[[ 2.20983545e-03  9.51934140e-03  5.41285099e-03 ... -2.79155909e-03
        -9.01016127e-03  1.36778364e-02]
       [-9.36647598e-03 -1.73871801e-03  1.08188950e-02 ...  1.51296274e-03
         7.57558912e-04  5.44300769e-03]
       [ 1.13182161e-02  4.26522718e-04  1.49796065e-03 ... -3.42557859e-03
        -1.79516233e-03 -4.79884632e-03]]]


     [[[-4.98654880e-03 -2.08284725e-02  2.18074489e-02 ... -7.44788814e-03
        -4.96819383e-03  1.28578511e-03]
       [ 1.82666897e-03  1.37834558e-02 -4.40165540e-03 ... -8.17192066e-03
         9.86121874e-03  1.46957850e-02]
       [ 1.46543132e-02  7.15820771e-03  5.12730749e-03 ... -1.95071772e-02
        -1.01376360e-03  6.32047653e-03]]]
```

步骤 10　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()
```

步骤 11　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：

```
epoch: 1 step: 596, loss is 0.023292765
epoch time: 36818.071 ms, per step time: 61.775 ms
epoch: 2 step: 596, loss is 0.0016317479
epoch time: 4320.621 ms, per step time: 7.249 ms
epoch: 3 step: 596, loss is 0.00025401893
epoch time: 4271.667 ms, per step time: 7.167 ms
epoch: 4 step: 596, loss is 0.0001774891
epoch time: 4332.598 ms, per step time: 7.269 ms
train success
```

步骤 12　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)
# 定义评估损失、网络
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()})
```

输出：

```
load checkpoint from [./ckpt/train_textcnn-4_596.ckpt].
accuracy:  {'acc': 0.7509765625}
```

步骤 13　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('o','')\
```

```
                                                .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):
    ##########请将代码补充完整#############
```

取单条评论文本数据，进行测试，输出情感类别及其概率

测试正面样例

输入：

```
review_en = "boring"
inference(review_en)
```

输出：

```
Negative comments
```

2.4 实验小结

本实验介绍了如何使用 MindSpore 搭建用于文本分类的 CNN 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。