



# 油饼味道不队

陈	王	子
温	家	伟
赵	航	怡

# 一、平台的安装

软件名称	版本	作用
Ubuntu 18.04/CentOS 7.6/EulerOS 2.8/openEuler 20.03/KylinV10 SP1	-	编译和运行MindSpore的操作系统
Python	3.7-3.9	MindSpore的使用依赖Python环境
昇腾AI处理器配套软件包	-	MindSpore使用的Ascend平台AI计算库
GCC	7.3.0	用于编译MindSpore的C++编译器

```
pip install https://ms-release.obs.cn-north-4.myhuaweicloud.com/2.1.1/MindSpore/unified/aarch64/mindspore-2.1.1-cp37-cp37m-linux_aarch64.whl --trusted-host ms-release.obs.cn-north-4.myhuaweicloud.com -i https://pypi.tuna.tsinghua.edu.cn/simple
```

## 二、平台的调试

```
seed@VM: ~/.../code3
import jieba
import mindspore.dataset as ds

# 用户输入的测试句子
text = "\"油饼味道不队\"由三人组成，立志于达到文本分析挖掘课程满分的成就。"

# 分词操作
tokens = jieba.lcut(text)

# 转换成 MindSpore Dataset 支持的形式
dataset = ds.NumpySlicesDataset([tokens], column_names=["text"])

# 输出分词结果
for data in dataset.create_dict_iterator():
    print(data["text"])

~
~
~
~
~
~
~
```

"test.py" 16L, 429C

5,18-14

All

Ubuntu测试代码

## 二、平台的调试

```
(MindSporepython==3.7.5) [10/10/23] seed@VM:~/.../code3$ python3 test.py
Building prefix dict from the default dictionary ...
Loading model from cache /tmp/jieba.cache
Loading model cost 0.559 seconds.
Prefix dict has been built successfully.
['"' '油饼' '味道' '不队' '"' '由三人' '组成' ', ' '立志' '于' '达到' '文本' '分
析' '挖掘' '课程'
'满分' '的' '成就' '。']
(MindSporepython==3.7.5) [10/10/23] seed@VM:~/.../code3$
```

分词测试成功

# 问题一：Python代码总被Kill

```
(MindSporepython==3.7.5) [10/12/23] seed@VM:~/.../code7$ vim test.py  
(MindSporepython==3.7.5) [10/12/23] seed@VM:~/.../code7$ python3 test.py  
Killed  
(MindSporepython==3.7.5) [10/12/23] seed@VM:~/.../code7$ python3 test.py  
Killed  
(MindSporepython==3.7.5) [10/12/23] seed@VM:~/.../code7$ python3 test.py
```

解决措施：

增加虚拟机内存，由1G增加到12G



## 问题二、属性缺失

加载预训练词向量  
数据集预处理完成  
模型构建完成

Traceback (most recent call last):

File "test.py", line 419, in <module>

grad fn = ms.value and grad(forward fn, None, optimizer.parameters)

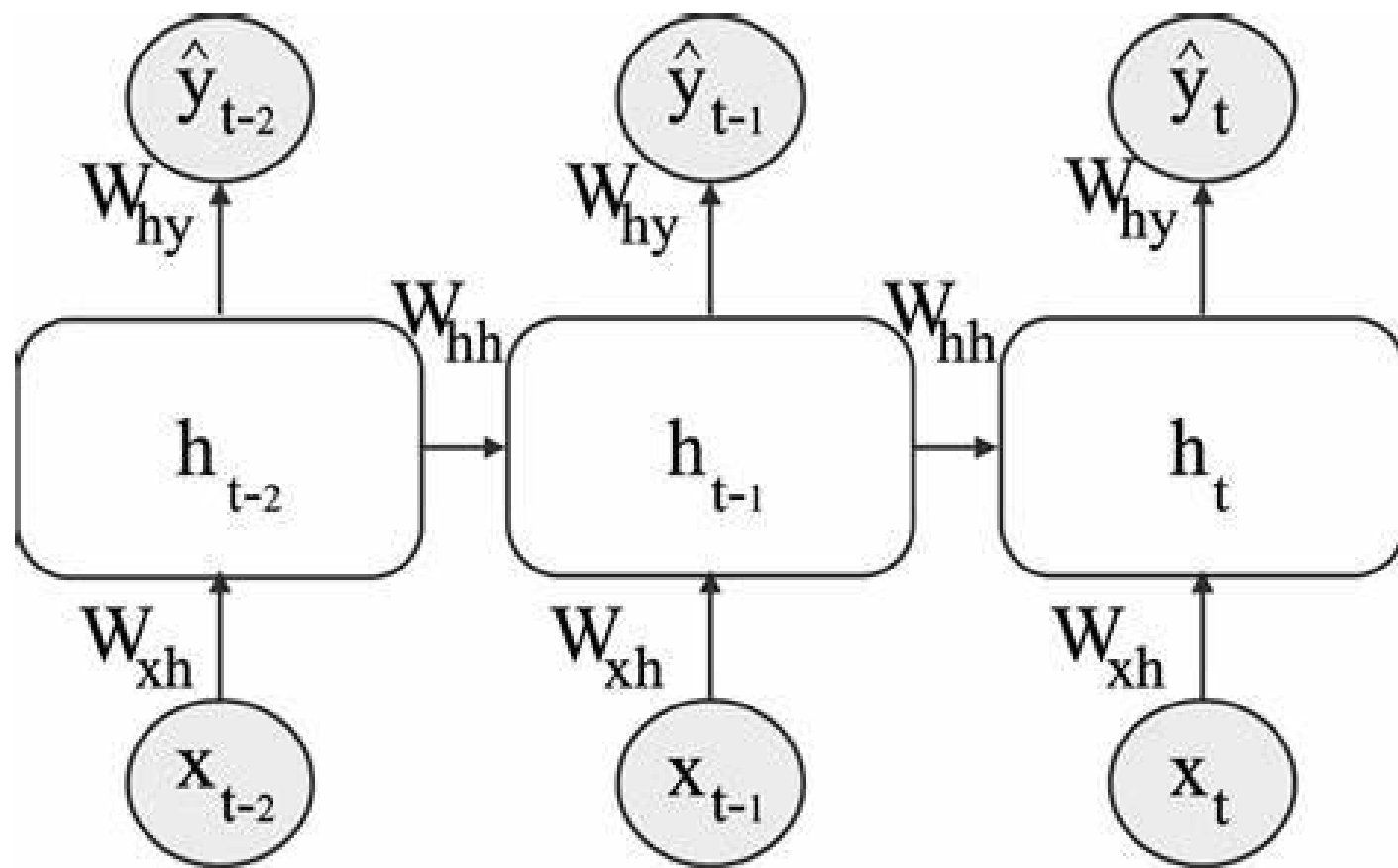
AttributeError: module 'mindspore' has no attribute 'value\_and\_grad'

(MindSporepython==3.7.5) [10/15/23] seed@VM:~/.../code/\$

更新Mindspore后解决

# 情感分类的原理

实现了基于RNN的情感分析模型的训练过程。通过定义模型结构、损失函数和优化器，以及对模型参数的更新和性能评估。通过多个epoch的迭代训练，逐渐提升模型的性能。



步骤	描述
定义RNN模型	定义一个RNN模型，包括LSTM层、词嵌入层和全连接层。
前向传播函数	实现RNN模型的前向传播逻辑，包括词嵌入、LSTM、全连接层的处理。
定义损失函数和优化器	采用二分类任务的损失函数BCEWithLogitsLoss，使用Adam优化器。
训练函数	定义训练步骤train_step和一个epoch的训练函数train_one_epoch。
评估函数	定义在测试集上评估模型性能的函数evaluate。
模型训练过程	设定总epoch数目，循环调用train_one_epoch和evaluate进行训练和评估。



# 三、训练情感分类模型

```
num_epochs = 5
best_valid_loss = float('inf')
ckpt_file_name = os.path.join(cache_dir, 'sentiment-analysis.ckpt')

for epoch in range(num_epochs):
    train_one_epoch(model, imdb_train, epoch)
    valid_loss = evaluate(model, imdb_valid, loss_fn, epoch)

    if valid_loss < best_valid_loss:
        best_valid_loss = valid_loss
        ms.save_checkpoint(model, ckpt_file_name)
```

[20]

```
... Epoch 0: 100%|██████████| 273/273 [08:23<00:00, 1.85s/it, loss=0.538]
Epoch 0: 100%|██████████| 117/117 [01:08<00:00, 1.70it/s, acc=0.709, loss=0.606]
Epoch 1: 100%|██████████| 273/273 [08:20<00:00, 1.83s/it, loss=0.57]
Epoch 1: 100%|██████████| 117/117 [01:08<00:00, 1.70it/s, acc=0.712, loss=0.6]
Epoch 2: 100%|██████████| 273/273 [08:18<00:00, 1.83s/it, loss=0.565]
Epoch 2: 100%|██████████| 117/117 [01:08<00:00, 1.70it/s, acc=0.714, loss=0.599]
Epoch 3: 100%|██████████| 273/273 [08:17<00:00, 1.82s/it, loss=0.57]
Epoch 3: 100%|██████████| 117/117 [01:08<00:00, 1.70it/s, acc=0.704, loss=0.609]
Epoch 4: 100%|██████████| 273/273 [08:19<00:00, 1.83s/it, loss=0.565]
Epoch 4: 100%|██████████| 117/117 [01:08<00:00, 1.71it/s, acc=0.722, loss=0.591]
```

可以看到每轮Loss逐步下降，在验证集上的准确率逐步提升。

```
param_dict = ms.load_checkpoint(ckpt_file_name)
ms.load_param_into_net(model, param_dict)
```

([], [])

```
imdb_test = imdb_test.batch(64)
evaluate(model, imdb_test, loss_fn)
```

Epoch 0: 100%|██████████| 391/391 [03:38<00:00, 1.79it/s, acc=0.5, loss=0.787]

0.7872264546811428

```
score_map = {
    1: "Positive",
    0: "Negative"
}
```

```
def predict_sentiment(model, vocab, sentence):
    model.set_train(False)
    tokenized = sentence.lower().split()
    indexed = vocab.tokens_to_ids(tokenized)
    tensor = ms.Tensor(indexed, ms.int32)
    tensor = tensor.expand_dims(0)
    prediction = model(tensor)
    return score_map[int(np.round(ops.sigmoid(prediction).asnumpy()))]
```

3]

## 四、模型测试

```
score_map = {
    1: "Positive",
    0: "Negative"
}

def predict_sentiment(model, vocab, sentence):
    model.set_train(False)
    tokenized = sentence.lower().split()
    indexed = vocab.tokens_to_ids(tokenized)
    tensor = ms.Tensor(indexed, ms.int32)
    tensor = tensor.expand_dims(0)
    prediction = model(tensor)
    return score_map[int(np.round(ops.sigmoid(prediction).asnumpy()))]
```

## 四、模型测试

```
predict_sentiment(model, vocab, "This film is terrible and bad")
```

'Negative'

```
predict_sentiment(model, vocab, "ZJUT is a terrible school")
```

'Negative'

```
predict_sentiment(model, vocab, "This film is great")
```

'Positive'

```
predict_sentiment(model, vocab, "ZJUT Tom Group is good team")
```

'Positive'

# 对比：Mindspore与Pytorch的不同

步骤	PyTorch	MindSpore
网络定义	继承 <code>nn.Module</code> ，在 <code>Net()</code> 中定义前向网络、损失函数和优化器	继承 <code>nn.Cell</code> ，在网络中定义前向网络，使用内置或自定义的损失函数和优化器
正向计算	运行实例化后的网络，得到logit，计算loss	运行实例化后的网络，得到logit，计算loss
反向计算	使用 <code>loss.backward()</code> 计算梯度	使用 <code>mindspore.grad()</code> 定义反向传播方程，将输入传入计算梯度
梯度更新	使用 <code>optim.step()</code> 将梯度更新到网络的Parameters	将Parameter的梯度传入定义好的optimizer中，完成梯度更新



# 另外的例子：机器翻译模型

步骤	描述
1. 定义NLLLoss类	计算负对数似然损失。
2. 定义WithLossCell类	构建带有损失函数的神经网络模型，使用Seq2Seq模型生成的encoder-decoder网络，计算图中包括将输出结果与标签计算损失的过程。
3. 创建Seq2Seq模型对象	使用WithLossCell封装为训练网络，并使用Adam优化器进行参数更新。
4. 定义回调函数（LossMonitor、ModelCheckpoint和TimeMonitor）	用于监控训练过程并保存模型。
5. 使用Model的train方法进行模型训练	执行模型训练。
6. 定义InferCell类	构建推理网络，调用Seq2Seq模型生成的encoder-decoder网络。
7. 加载训练好的模型参数，并构建推理模型	加载已经训练好的模型参数，并构建推理模型。
8. 定义translate函数	将英文句子翻译为中文。首先将英文句子转换为索引表示，然后通过推理模型生成中文句子。

我们实现了Seq2Seq模型的训练和推理功能，通过定义损失函数、构建模型、进行训练和推理，实现了将英文句子翻译为中文的功能。



# 机器翻译模型演示

```
class Seq2Seq(nn.Cell):
    def __init__(self, config, is_train=True):
        super(Seq2Seq, self).__init__()
        self.max_len = config.max_seq_length
        self.is_train = is_train

        self.encoder = Encoder(config, is_train)
        self.decoder = Decoder(config, is_train)
        self.expanddims = P.ExpandDims()
        self.squeeze = P.Squeeze(axis=0)
        self.argmax = P.ArgMaxWithValue(axis=int(2), keep_dims=True)
        self.concat = P.Concat(axis=1)
        self.concat2 = P.Concat(axis=0)
        self.select = P.Select()

    def construct(self, src, dst):
        encoder_output, hidden = self.encoder(src)
        decoder_hidden = self.squeeze(encoder_output[self.max_len-2:self.max_len-1:1, ::, ::])
        if self.is_train:
            outputs, _ = self.decoder(dst, decoder_hidden, encoder_output)
        else:
            decoder_input = dst[:, 0:1:1]
            decoder_outputs = ()
            for i in range(0, self.max_len):
                decoder_output, decoder_hidden, _ = self.decoder(decoder_input,
                                                                    decoder_hidden, encoder_output)
                decoder_hidden = self.squeeze(decoder_hidden)
                decoder_output, _ = self.argmax(decoder_output)
                decoder_output = self.squeeze(decoder_output)
                decoder_outputs += (decoder_output,)
                decoder_input = decoder_output
            outputs = self.concat(decoder_outputs)
        return outputs

print("11.定义Seq2Seq整体结构成功")
```

```
def translate(str_en):
    max_seq_len = 10
    str_vocab = normalizeString(str_en).split(' ')
    print("English", str(str_vocab))
    str_id = [1]
    for i in str_vocab:
        str_id += [en_vocab.index(i)]

    num = max_seq_len + 1 - len(str_id)
    if(num >= 0):
        str_id += [0]*num
    else:
        str_id = str_id[:max_seq_len] + [0]
    str_id = Tensor(np.array([str_id[1:]]).astype(np.int32))

    out_id = [1]+[0]*10
    out_id = Tensor(np.array([out_id[:-1]]).astype(np.int32))

    output = network(str_id, out_id)
    out = ''
    for x in output[0].asnumpy():
        if x == 0:
            break
        out += ch_vocab[x]
    print("中文", out)

print("17.定义翻译测试函数成功")
translate('ZJUT LOL Group did great job.')
translate('teacher Mei is the best teacher.')
```

17.定义翻译测试函数成功

English ZJUT Tom Group did great job.


中文 浙江工业大学汤姆队伍做得很好.

English teacher Mei is the best teacher.

中文 梅老师是最好的老师.




# 分工情况

- 温家伟
    - 参与项目选题讨论，环境依赖安装、代码调试、运行、模型训练与测试、PPT训练与测试部分的撰写
    - （主代码）
  - 赵挺钧
    - 参与项目选题讨论，环境依赖安装、代码调试运行、模型训练与测试、PPT调试问题部分撰写
    - （主代码）
  - 陈王子
    - 参与项目选题讨论，环境依赖安装、代码调试运行、模型训练与测试、PPT原理部分撰写、课堂汇报
    - （主统筹）
- 



# 问题和总结

- 1.运行内存不够导致中断
  - 使用华为云进行操作
  - 2.学习代码mindspore版本过旧
  - 修改代码匹配新版mindspore
  - 3.loss下降得值太小
  - 适当改变学习率和增大学习次数
- 

The background features a light gray base with several abstract shapes. In the top left, there's a green circle with diagonal lines and a solid light blue shape. In the top right, a large blue circle with diagonal lines overlaps a solid green shape. The bottom left has a large blue shape and a yellow circle with diagonal lines. The bottom right features a green wavy shape and a blue wavy shape. The text "谢谢大家!" is centered in the middle of the image.

谢谢大家!