

# 《自然语言处理》

## 基于华为云的 命名实体识别实验指导

哈尔滨工业大学  
自然语言处理课程组  
2021 年秋季学期





# 目录

---

<b>1 实验总览</b>	<b>2</b>
1.1 实验背景	2
1.2 实验目的	2
1.3 实验清单	2
<b>2 命名实体识别实验</b>	<b>3</b>
2.1 实验简介	3
2.2 实验环境	3
2.3 背景知识	3
2.4 实验步骤	4
2.4.1 实验准备	4
2.4.2 实验过程	8
2.5 实验总结	35



# 1 实验总览

## 1.1 实验背景

命名实体识别 (Named Entity Recognition, NER) 是 NLP 领域最经典的任务之一，实体识别提取一些专有的实体，如人名，地名，机构名，公司名，药品名等，实体识别广泛应用于搜索，对话，问答，知识库构建等场景中。

## 1.2 实验目的

本实验的主要目的是掌握命名实体识别 (NER) 相关基础知识点，使用开源工具以及 MindSpore 框架实现命名实体识别模型，加深对相关理论的理解。

## 1.3 实验清单

表格：实验、简述、难度、软件环境、硬件环境。

实验	简述	难度	软件环境	开发环境
命名实体识别实验 (BERT+CRF)	使用BERT+CRF实现 命名实体识别	高级	Python3.7、 MindSpore1.3	ModelArts GPU Notebook环境、 ModelArts训练 作业

表 1.1 实验清单表格



# 2 命名实体识别实验

## 2.1 实验简介

命名实体识别 (Named Entity Recognition, NER) 是 NLP 领域最经典的任务之一，实体识别提取一些专有的实体，如人名，地名，机构名，公司名，药品名等，实体识别广泛应用于搜索，对话，问答，知识库构建等场景中。基于 transformer 的 BERT 预训练模型相对于循环神经网络 (Recurrent Neural Network, RNN)，长短期记忆网络 (Long Short-Term Memory, LSTM) 以及传统的隐马尔科夫模型 (Hidden Markov Model, HMM)、条件随机场 (Conditional Random Field, CRF) 能够更好地捕捉上下文语义，从而提升识别性能。

本实验在华为云 ModelArts 平台上使用 MindSpore1.3 实现 BERT+CRF 命名实体识别模型。

## 2.2 实验环境

ModelArts GPU Notebook 环境、ModelArts 训练作业。

## 2.3 背景知识

命名实体识别不仅要找出实体的位置，还要对实体进行分类。位置和类别通过标签来表达，命名实体识别数据标注格式有 BIO 和 BIOES 两种：

	BIO	BIOES
小	B-PER	B-PER
强	I-PER	E-PER
去	O	O
培	B-ORG	B-ORG
训	I-ORG	I-ORG
中	I-ORG	I-ORG
心	I-ORG	E-ORG
学	O	O
习	O	O

图 2.1 命名实体识别数据标注格式

上面这个例子中 BIO 或 BIOES 代表实体的位置，B 代表 begin, I 代表 inner, O 代表 other, E 代表 end, S 代表 single(表示这个实体只有一个词)，PER 或 ORG 代表实体的类别，PER 代表 Person(人名)，ORG 代表 Organization (机构名)。

BERT 做命名实体识别的基本原理：在每一个输入对应的位置上输出，做一个多分类，得到对应的标签。

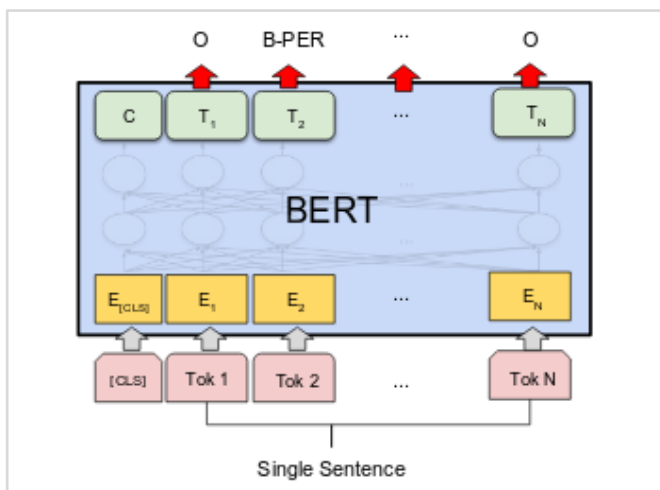


图 2.2 BERT 命名实体识别

## 2.4 实验步骤

### 2.4.1 实验准备

#### 步骤 1 登录 OBS

注册并登录华为云官网（<https://www.huaweicloud.com/>），点击右上角的控制台，如图所示



图 2.3 华为云官网

点击左上角的按钮，打开控制台服务列表

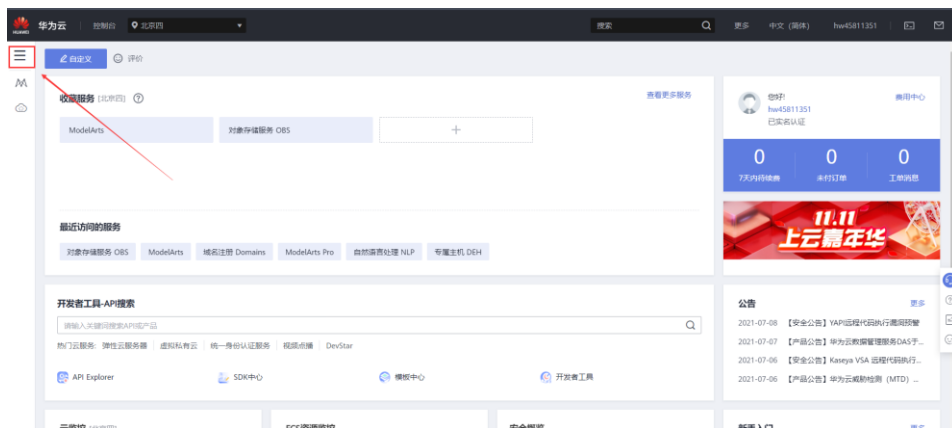


图 2.4 控制台界面



## 点击 OBS 服务

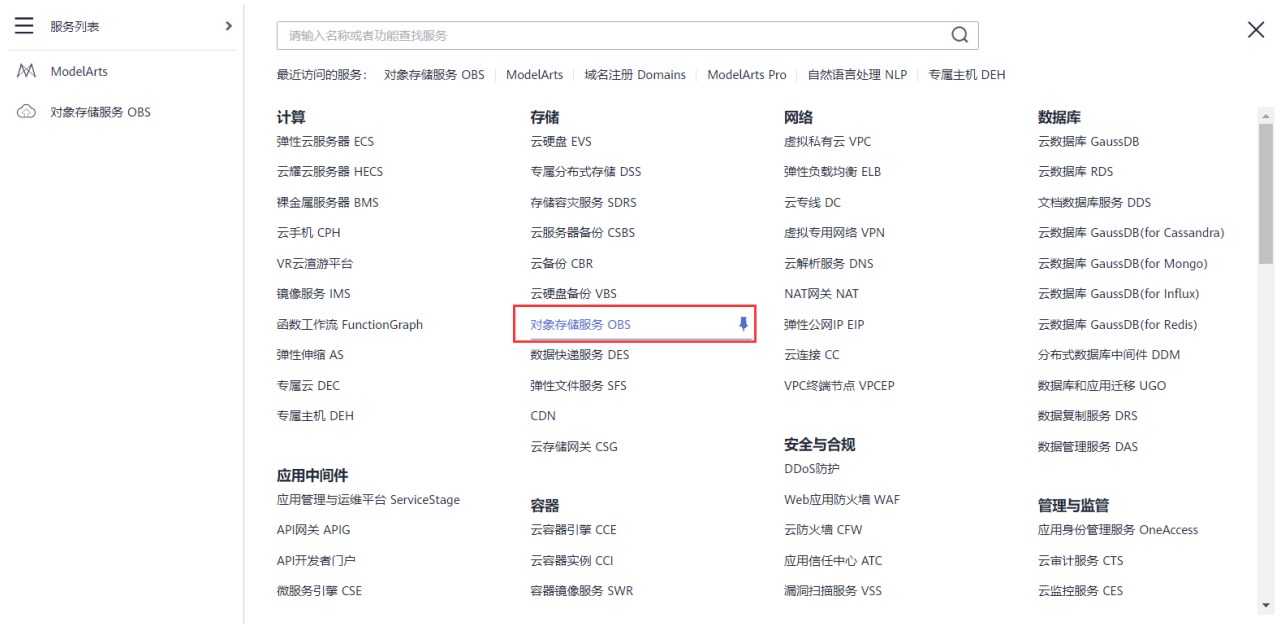


图 2.5 服务列表

## 步骤 2 创建 OBS 桶

## 点击创建桶

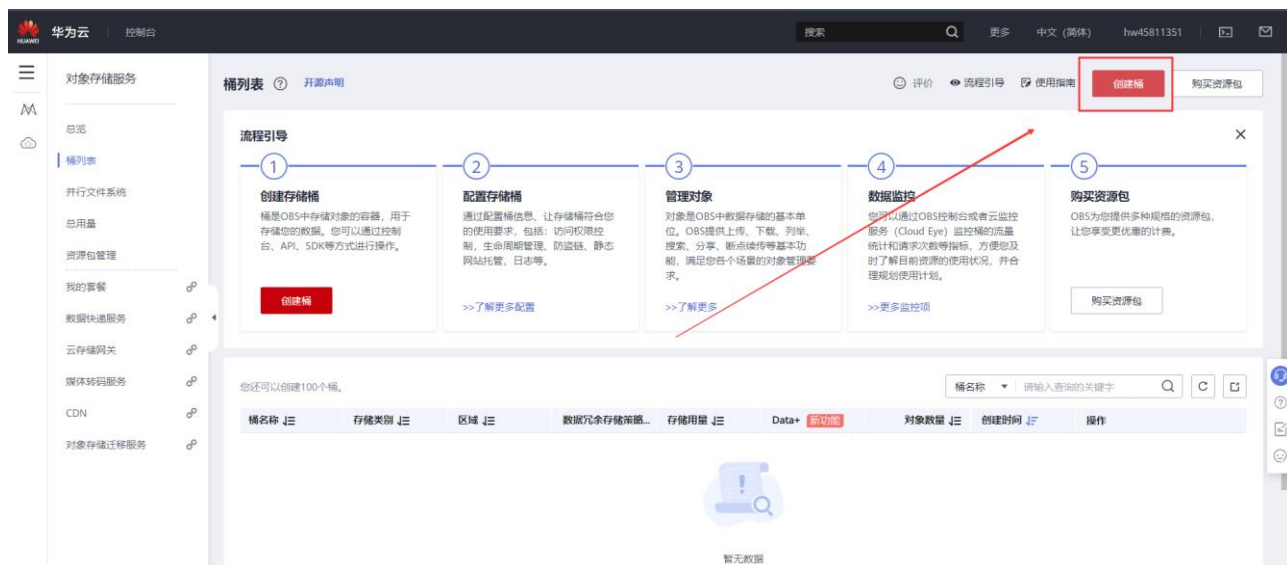


图 2.6 OBS 界面

复制桶配置不用选择，区域选择“华北-北京四”，桶名称可自定义，数据冗余存储策略选择“单 AZ 存储”，存储类别选择“标准存储”，桶策略选择“私有”，默认加密勾选“开启”同时创建密钥并选择（也可不勾选，则无需密钥），归档存储数据直读选择“关闭”，标签项无需填写。



复制桶配置

选择源桶

该项可选。选择后可复制源桶的以下配置信息：区域 / 数据冗余策略 / 存储类别 / 桶策略 / 默认加密 / 归档数据直读 / 企业项目 / 标签。

区域

华北-北京四

已有资源包区域 华北-北京四

不同区域的云服务产品之间内网互不相通；请就近选择靠近您业务的区域，可减少网络时延，提高访问速度。[如何选择区域](#)

桶名称

season-ner

不能和本用户已有桶重名

不能和其他用户已有的桶重名

创建成功后不支持修改

已购存储包

标准存储包 (单AZ) 剩余: 40 GB

可参考当前区域已购存储包，创建相应类型的桶。

数据冗余存储策略

多AZ存储

单AZ存储

启用后不支持修改

数据在同区域的多个AZ中存储，可用性更高。

默认存储类别

标准存储

适合高性能，高可靠，高可用，频繁访问场景。

多AZ存储

单AZ存储

图片处理

成本视图

存储

高

数据取回

无

请求次数

中

低频访问存储

适合高可靠，低成本，较少访问场景

多AZ存储

单AZ存储

图片处理

成本视图

存储

中

数据取回

低

请求次数

低

归档存储

适合长期存储，基本不访问场景

单AZ存储

成本视图

存储

低

数据取回

高

请求次数

低

创建桶时选择的存储类别会作为上传对象的默认存储类别。[了解存储类别差异](#)

桶策略

私有

公共读

公共读写

复制桶策略

桶的拥有者拥有完全控制权限，其他用户在未经授权的情况下均无访问权限。

默认加密

开启默认加密

免费

建议开启默认加密，密钥管理全免费，核心数据更安全。

开启默认加密后，上传对象或文件时将默认采用以下密钥进行加密。

KMS-37d9

创建KMS密钥

归档数据直读

开启

关闭

关闭归档直读，归档存储类别的数据要先恢复才能访问。归档存储数据恢复和访问会收取相应的费用。[价格详情](#)

标签

如果您需要使用同一标签标识多种云资源，即所有服务均可在标签输入框下拉选择同一标签，建议在TMS中创建预定义标签。[查看预定义标签](#)

标签键

标签值

您还可以添加10个标签。

存储包超值购

您在当前区域下已有标准存储包 (单AZ) 剩余: 40 GB，请根据资源包余量评估是否需要加购存储资源包。

图 2.7 OBS 桶配置

点击“立即创建”即可，创建成功如下。

桶列表

您还可以创建99个桶。

桶名称

季

存储类别

季

区域

季

数据冗余存储策略...

季

存储用量

季

Data+

新功能

对象数量

季

创建时间

季

操作

season-ner

--

华北-北京四

--

--

--

2021/11/10 18:47...

修改存储类别

删除

图 2.8 OBS 桶列表

### 步骤 3 OBS 上传对象

点击创建的 OBS 桶，点击左侧的“对象”。

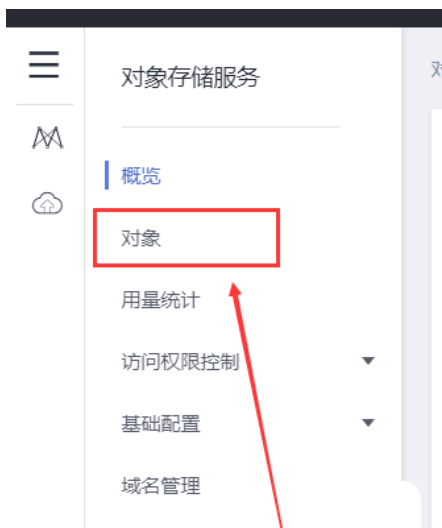


图 2.8 OBS 桶左侧

点击“上传对象”。



图 2.9 OBS 桶对象

将实验目录 bert（请从 QQ 群中下载 bert.zip，然后解压）拖拽至上传区，点击“上传”即可。

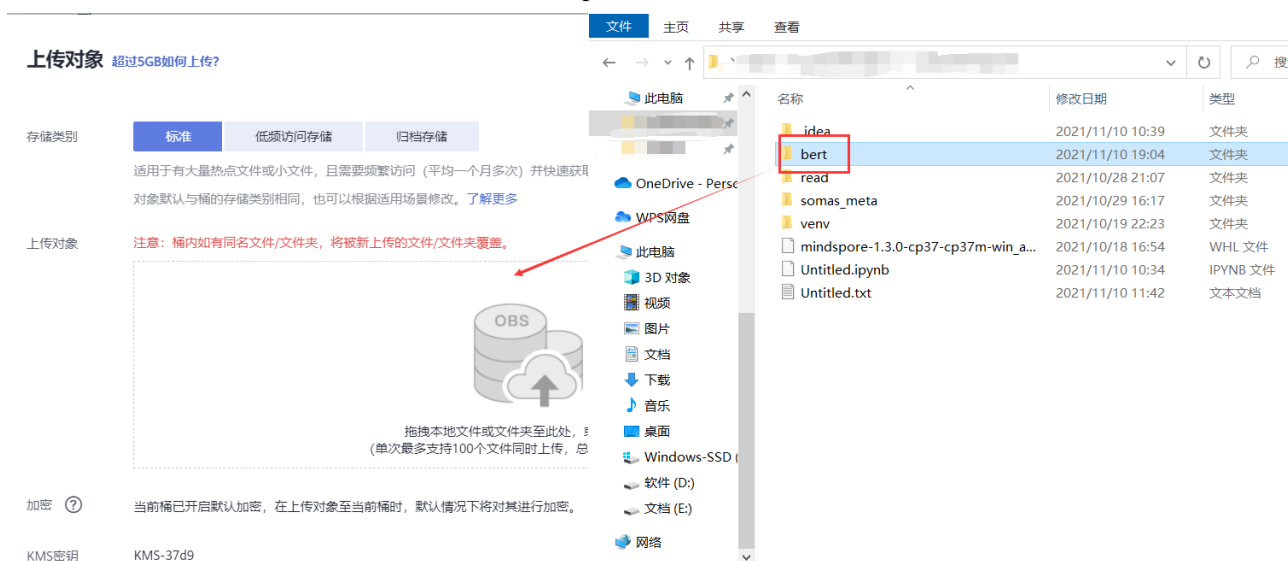


图 2.10 上传对象





点击下方的任务管理，可以查看上传进度。

## 任务管理

✕

[正在运行](#)[已完成](#)[失败](#)

删除所有

运行所有

取消所有

对象名称	桶名称	大小	类型	状态	操作
bert/pre_model/ber...	season-ner	1.15 GB	上传	<div></div>	<a href="#">取消</a> <a href="#">删除</a>
bert/data/train.tf_re...	season-ner	6.78 MB	上传	<div></div>	<a href="#">取消</a> <a href="#">删除</a>

关闭

图 2.11 上传进度

等待上传结束即可，过程约为几分钟，注意上传期间不要关闭标签页。

上传完成后，会出现如下界面

[对象](#)[已删除对象](#)[碎片](#)

对象是数据存储的基本单位，在OBS中文件和文件夹都是对象。您可以上传任何类型（文本、图片、视频等）的文件，并在桶中对这些文件进行管理。[了解更多](#)  
若需要将对象移动到桶内其他位置，推荐下载使用[OBS Browser+](#)图形化管理工具。

上传对象

新建文件夹

恢复

更多

输入对象名前缀搜索

Q

C

<input type="checkbox"/>	名称	存储类别	大小	加密状态	恢复状态	最后修改时间	操作
<input type="checkbox"/>	bert	--	--	--	--	--	<a href="#">分享</a> <a href="#">复制路径</a> <a href="#">更多</a>

图 2.12 上传结果

点击文件夹，如下

[对象](#)[已删除对象](#)[碎片](#)

对象是数据存储的基本单位，在OBS中文件和文件夹都是对象。您可以上传任何类型（文本、图片、视频等）的文件，并在桶中对这些文件进行管理。[了解更多](#)  
若需要将对象移动到桶内其他位置，推荐下载使用[OBS Browser+](#)图形化管理工具。

上传对象

新建文件夹

恢复

更多

输入对象名前缀搜索

Q

C

<input type="checkbox"/>	名称	存储类别	大小	加密状态	恢复状态	最后修改时间	操作
<a href="#">返回上一级</a>							
<input type="checkbox"/>	code	--	--	--	--	--	<a href="#">分享</a> <a href="#">复制路径</a> <a href="#">更多</a>
<input type="checkbox"/>	data	--	--	--	--	--	<a href="#">分享</a> <a href="#">复制路径</a> <a href="#">更多</a>
<input type="checkbox"/>	eval_out	--	--	--	--	--	<a href="#">分享</a> <a href="#">复制路径</a> <a href="#">更多</a>
<input type="checkbox"/>	model_finete...	--	--	--	--	--	<a href="#">分享</a> <a href="#">复制路径</a> <a href="#">更多</a>
<input type="checkbox"/>	pre_model	--	--	--	--	--	<a href="#">分享</a> <a href="#">复制路径</a> <a href="#">更多</a>
<input type="checkbox"/>	env.sh	标准存储	852 bytes	已加密	--	2021/11/10 21:18:49 G...	<a href="#">下载</a> <a href="#">分享</a> <a href="#">更多</a>

图 2.13 OBS 桶下 bert 文件夹

即为上传成功。

## 2.4.2 实验过程

本次实验可采用 ModelArts 中的 Notebook 或训练作业完成。



进入 ModelArts 流程如下：

在控制台服务列表中点击 ModelArts。

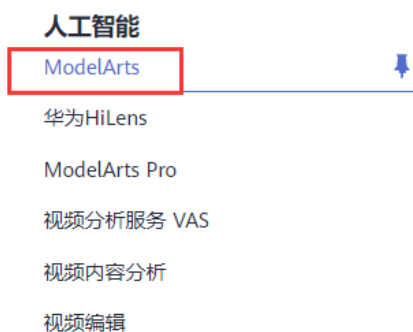


图 2.14 服务列表

左上角服务器选择“华北-北京四”。



图 2.15 服务器选择

## 2.4.2.1 Notebook 实验

### 步骤 1 新建 Notebook

点击左侧“开发环境”下的“Notebook”，如下

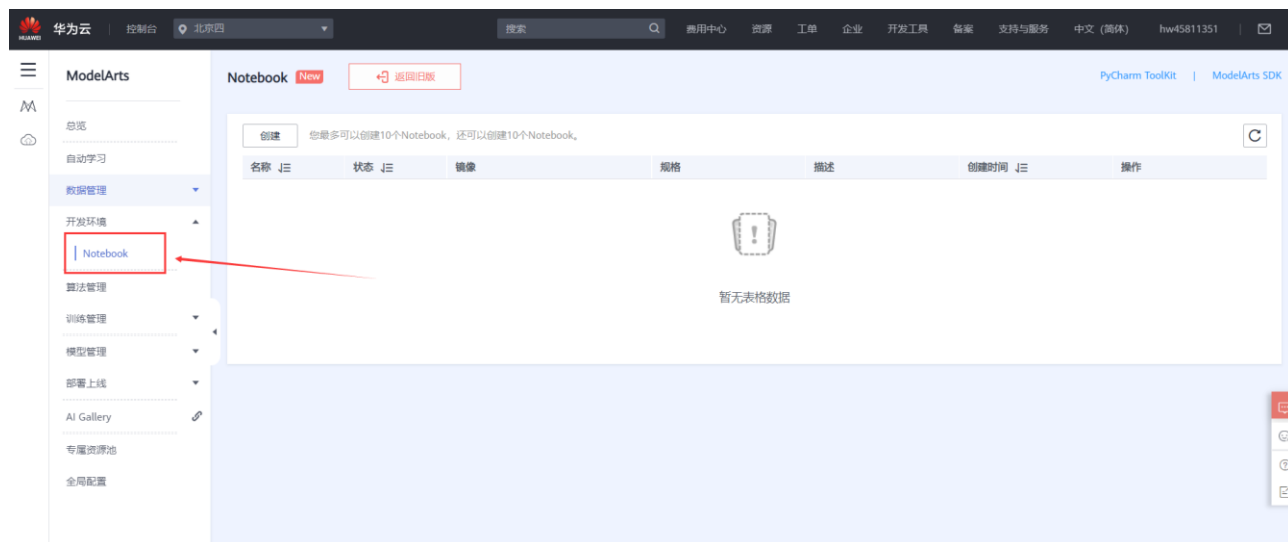


图 2.16 ModelArts 界面



点击上方的“返回旧版”



图 2.17 Notebook 列表界面

点击“创建”



图 2.18 旧版 Notebook 界面

名称自定义，工作环境选择“Multi-Engine 2.0 (Python3)”



图 2.19 环境选择

资源类型选择“GPU”，规格选择“限时免费”，并勾选“我已同意”



图 2.20 资源选择



存储位置选择“对象存储服务（OBS）”，点击右下方的“选择”

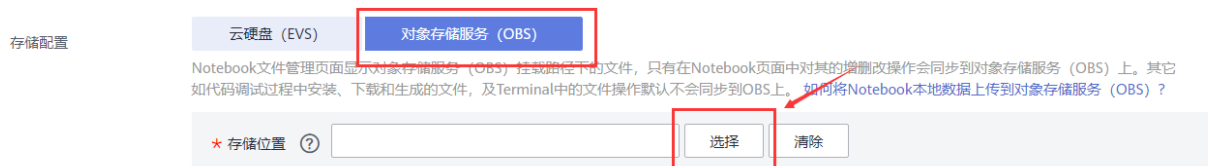


图 2.21 存储配置

点击刚才新建的桶

## 存储位置

请选择文件夹。

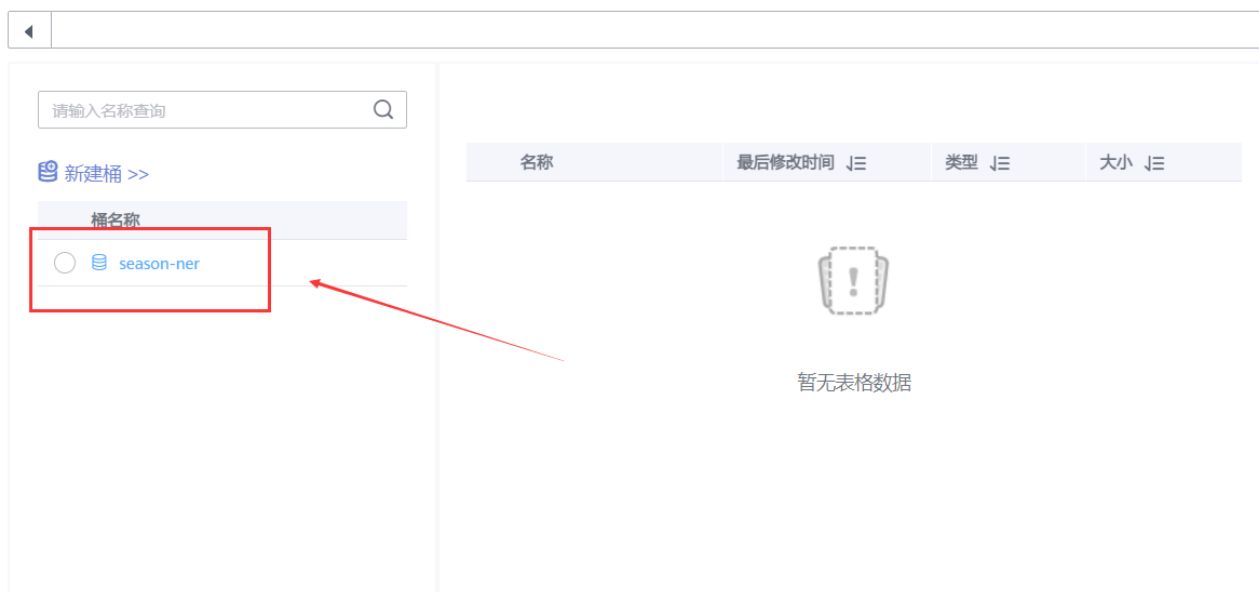


图 2.22 存储位置选择-选择桶

点击 bert 文件夹前的小圆圈

## 存储位置

请选择文件夹。

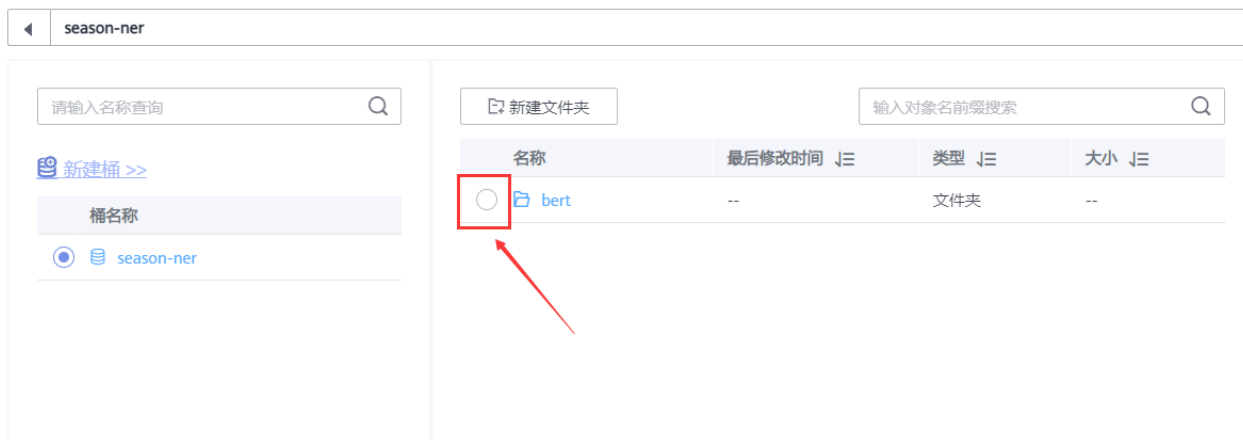


图 2.23 存储位置选择-选择文件夹

勾选如下，切记勾选小圆圈，而非打开该文件夹



## 存储位置

请选择文件夹。

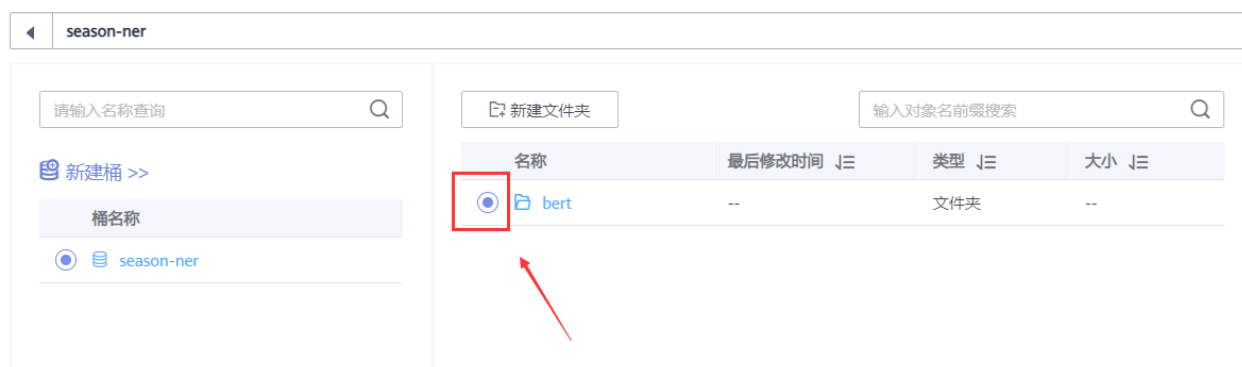


图 2.23 存储位置选择-选择后结果

点击下方的“确定”



图 2.24 点击确定

结果如下，存储位置为“/你的桶名/bert/”



图 2.25 存储位置选择结果

点击右下角的“下一步”

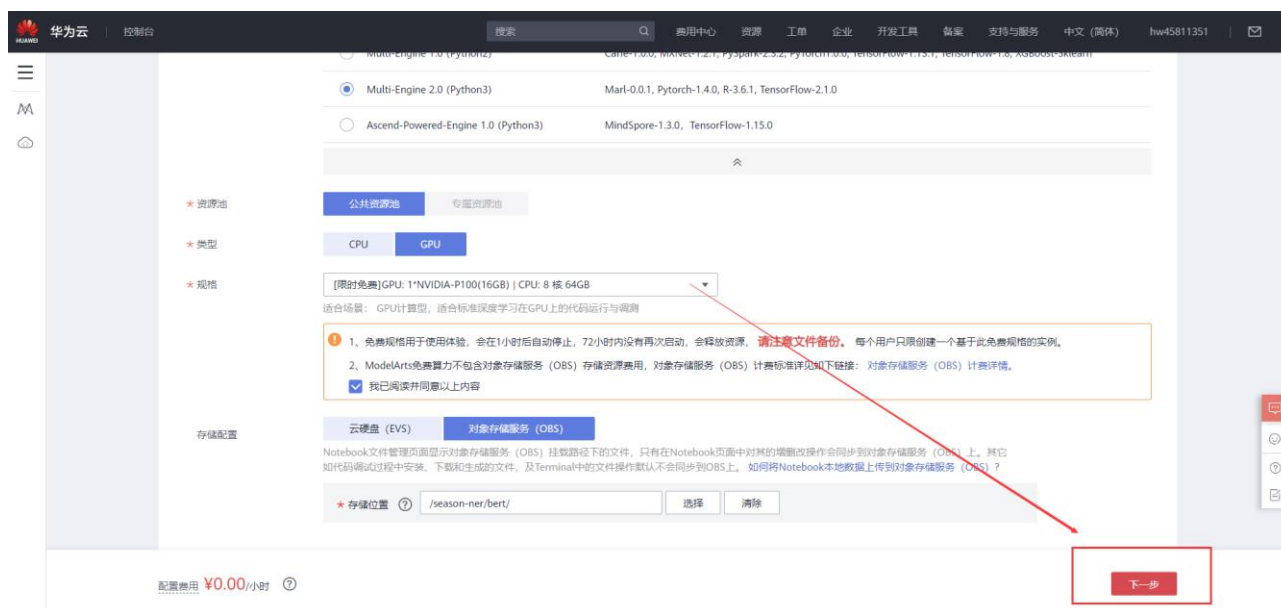


图 2.26 点击下一步



点击“提交”

产品名称	产品规格	计费模式	价格
mindspore-ner	自动停止时间	1小时	
	描述	--	
	工作环境	Multi-Engine 2.0 (Python3)	
	资源池	公共资源池	
	类型	GPU	
	规格	[限时免费]GPU: 1*NVIDIA-P100(16GB)   CPU: 8 核 64GB	
	存储配置	OBS (/season-ner/bert/)	
		按需计费	Notebook: ¥0.00/小时

图 2.27 点击提交

结果如下

任务提交成功!

您的Notebook实例已开始创建, 请耐心等待。

返回 Notebook 列表

您的余额: ¥ 5.24 充值

图 2.28 提交结果

点击“返回 Notebook 列表”，有时会出现“资源排队”，则点击“资源排队”，等排到时再进行后续操作。

名称	状态	工作环境	规格	描述	创建时间	创建者	操作
mindspore-ner	运行中 (59 分钟后停止)	Multi-Engine ...	[限时免费]GPU: 1*...	--	2021/11/10 22:05...	hw45811351	打开 打开JupyterLab 停止 删除

图 2.29 Notebook 列表

## 步骤 2 同步代码

点击“打开”



图 2.30 点击打开

选中 code 文件夹和 env.sh 文件，再点击“Sync OBS”，弹出框选择“Yes”。

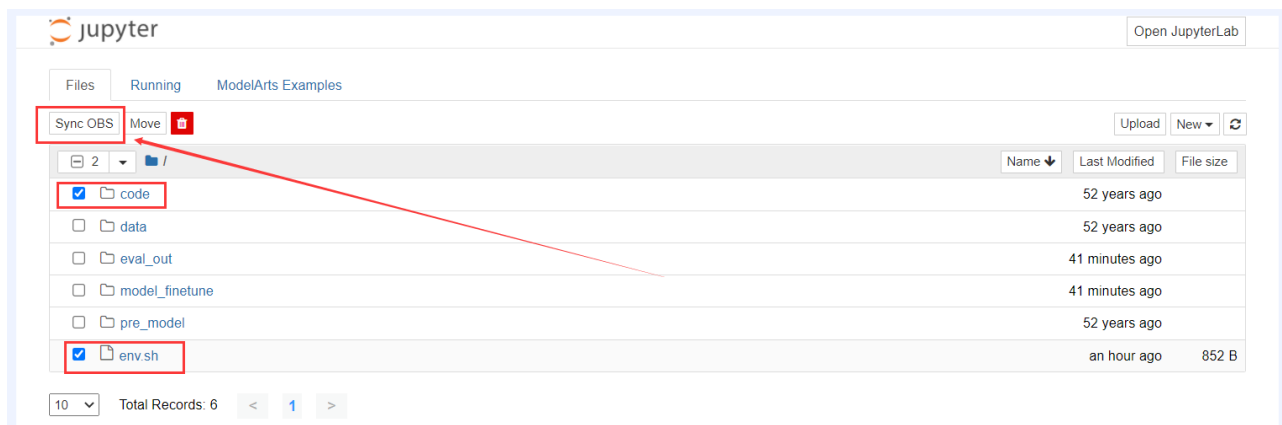


图 2.31 Jupyter 界面

### 步骤 3 配置环境

点击左上角“Notebook”返回 Notebook 列表

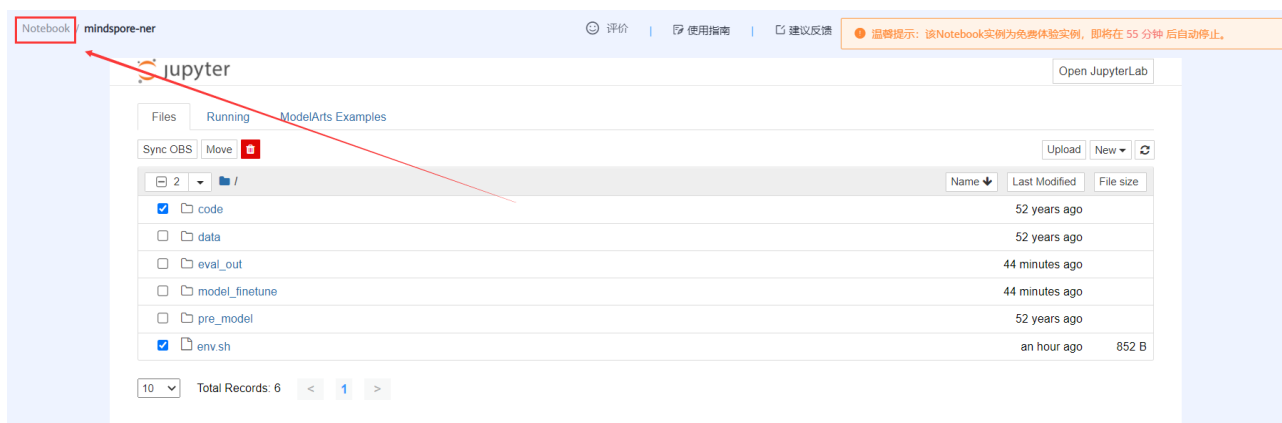


图 2.32 Jupyter 界面

点击“打开 JupyterLab”



图 2.33 打开 JupyterLab



点击“Terminal”

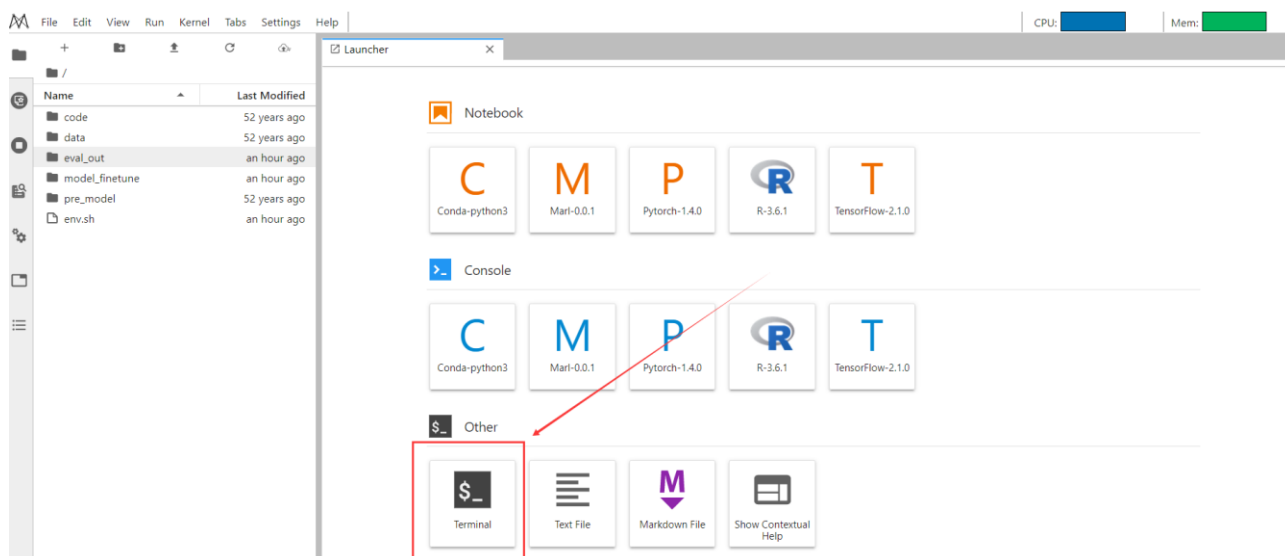


图 2.34 JupyterLab 界面

输入命令

```
cd work
ls
```

查看代码是否同步成功，出现如下结果，即为同步成功。

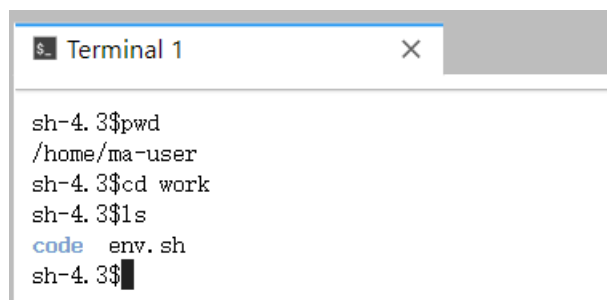


图 2.35 终端结果

运行配置脚本，输入命令

```
sh env.sh
```

```
sh-4.3$ sh env.sh
Looking in indexes: https://pypi.tuna.tsinghua.edu.cn/simple
Collecting mindspore-gpu==1.3.0
  Downloading https://ms-release.obs.cn-north-4.myhuaweicloud.com/1
    139.6 MB 4.8 kB/s
```

图 2.36 配置环境

等待配置结束（此处的 ERROR 可忽略）

```
Building wheels for collected packages: easydict, terminaltables
  Building wheel for easydict (setup.py) ... done
  Created wheel for easydict: filename=easydict-1.9-py3-none-any.whl size=6348 sha256=958086090ae69511440e14131748572b49aee6356506dbc98ad3646eb1c29a84
  Stored in directory: /home/ma-user/.cache/pip/wheels/36/14/48/d794ec91e2076042588d2a16f57f8b0b80b08146428848e47b
  Building wheel for terminaltables (setup.py) ... done
  Created wheel for terminaltables: filename=terminaltables-3.1.0-py3-none-any.whl size=15354 sha256=1de6d48c9a3306a7b022ae2c59cd68c1fc428c5d2ca3d32de8de3ad004dc9725
  Stored in directory: /home/ma-user/.cache/pip/wheels/47/fc/8b/50a18c51662c4c99326f26b8d644985006b049dd62194abc0a
Successfully built easydict terminaltables
Installing collected packages: yapf, terminaltables, PyYAML, opencv-python, ninja, lxml, enum34, easydict, addict, absl-py
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the following dependency conflicts.
moxing 2.0.0rc0 requires urllib3>1.26.2, but you have urllib3 1.24.1 which is incompatible.
Successfully installed PyYAML-5.4.1 absl-py-0.12.0 addict-2.4.0 easydict-1.9 enum34-1.1.10 lxml-4.6.3 ninja-1.10.0.post3 opencv-python-4.5.3.56 terminaltables-3.1.0 yapf-0.31.0
sh-4.3$
```

图 2.37 配置结果





#### 步骤 4 启动微调训练（无 CRF）

输入命令（此处为一行命令）

```
python code/main.py --device_target=GPU --data_url='s3://你的桶名/bert/data/' --ckpt_url='s3://你的桶名/bert/pre_model/' --train_url='s3://你的桶名/bert/model_finetune/'
```

```
sh-4.3$ python code/main.py --device_target=GPU --data_url='s3://season-ner/bert/data/' --ckpt_url='s3://season-ner/bert/pre_model/' --train_url='s3://season-ner/bert/model_finetune/'
```

图 2.38 命令示例

（训练过程中出现[WARNING]信息均可忽略）

若出现如下错误“Failed to set current device id”

```
RuntimeError: mindspore/ccsrc/runtime/device/gpu/gpu_device_manager.cc:27 InitDevice] Op Error: Failed to set current device id | Error Number: 0
```

图 2.39 device id 错误

则需返回 Notebook 列表，停止并删除该 Notebook，重新新建 Notebook，即重复步骤 1-4。



图 2.40 停止删除 Notebook

若出现以下内容，则表明微调训练成功运行，耐心等待最终训练结束即可（总时长约 30-40 分钟）。

```
sh-4.3$ python code/main.py --device_target=GPU --data_url='s3://season-ner/bert/data/' --ckpt_url='s3://season-ner/bert/pre_model/' --train_url='s3://season-ner/bert/model_finetune/'
python_version: 3.7.3 (default, Mar 27 2019, 22:11:17)
[GCC 7.3.0]
*****
[WARNING] ME (938:139688740505408,MainProcess):2021-11-10-22:43:47.417.690 [code/main.py:208] GPU only support fp32 temporarily, run with fp32.
INFO:root:Using MoXing-v2.0.0.rc0-19e4d3ab
INFO:root:Using OBS-Python-SDK-3.20.9.1
*****
data_size: 10748
*****
steps_per_epoch: 671
*****
[WARNING] ME (938:139688740505408,MainProcess):2021-11-10-22:44:26.904.299 [mindspore/train/serialization.py:559] 2 parameters in the net are not loaded.
[WARNING] ME (938:139688740505408,MainProcess):2021-11-10-22:44:26.904.628 [mindspore/train/serialization.py:561] bert.dense_l.weight is not loaded.
[WARNING] ME (938:139688740505408,MainProcess):2021-11-10-22:44:26.904.680 [mindspore/train/serialization.py:561] bert.dense_l.bias is not loaded.
[WARNING] CORE (938,7f0bd1c1c740,python):2021-11-10-22:45:58.562.112 [mindspore/core/ir/anf_extends.cc:62] fullname_with_scope Input 0 of cnode is not a value node, its type is CNode.
[WARNING] DEVICE (938,7f091ffff700,python):2021-11-10-22:45:59.649.200 [mindspore/ccsrc/runtime/device/kernel_runtime.cc:345] AssignStaticMemoryInput It is not suggested to use a lonely weight parameter as the output of graph
[WARNING] DEVICE (938,7f091ffff700,python):2021-11-10-22:45:59.649.249 [mindspore/ccsrc/runtime/device/kernel_runtime.cc:345] AssignStaticMemoryInput It is not suggested to use a lonely weight parameter as the output of graph
epoch: 0, current epoch percent: 1.000, step: 671, outputs are (Tensor(shape=[], dtype=Float32, value= 0.115073), Tensor(shape=[], dtype=Bool, value= False))
[WARNING] DEBUG (938,7f0bd1c1c740,python):2021-11-10-22:54:30.549.457 [mindspore/ccsrc/debug/dump_proto.cc:467] ExportCNode] Operator must be a primitive
[WARNING] DEBUG (938,7f0bd1c1c740,python):2021-11-10-22:54:30.550.647 [mindspore/ccsrc/debug/dump_proto.cc:231] SetValueToProto] Unsupported type FuncGraph
[WARNING] DEBUG (938,7f0bd1c1c740,python):2021-11-10-22:54:30.550.661 [mindspore/ccsrc/debug/dump_proto.cc:231] SetValueToProto] Unsupported type FuncGraph
epoch time: 605068.663 ms, per step time: 901.742 ms
```

图 2.41 epoch 信息

训练结束，如下所示

```
epoch: 0, current epoch percent: 1.000, step: 671, outputs are (Tensor(shape=[], dtype=Float32, value= 0.1122), Tensor(shape=[], dtype=Bool, value= False))
[WARNING] DEBUG (919,7f40a36a1740,python):2021-11-11-10:20:31.334.488 [mindspore/ccsrc/debug/dump_proto.cc:467] ExportCNode] Operator must be a primitive
[WARNING] DEBUG (919,7f40a36a1740,python):2021-11-11-10:20:31.335.734 [mindspore/ccsrc/debug/dump_proto.cc:231] SetValueToProto] Unsupported type FuncGraph
epoch time: 615375.294 ms, per step time: 917.102 ms
epoch: 1, current epoch percent: 1.000, step: 1342, outputs are (Tensor(shape=[], dtype=Float32, value= 0.082431), Tensor(shape=[], dtype=Bool, value= False))
epoch time: 508915.771 ms, per step time: 758.444 ms
epoch: 2, current epoch percent: 1.000, step: 2013, outputs are (Tensor(shape=[], dtype=Float32, value= 0.0609034), Tensor(shape=[], dtype=Bool, value= False))
epoch time: 511214.054 ms, per step time: 761.869 ms
epoch: 3, current epoch percent: 1.000, step: 2684, outputs are (Tensor(shape=[], dtype=Float32, value= 0.0362411), Tensor(shape=[], dtype=Bool, value= False))
epoch time: 512832.950 ms, per step time: 764.282 ms
epoch: 4, current epoch percent: 1.000, step: 3355, outputs are (Tensor(shape=[], dtype=Float32, value= 0.0308992), Tensor(shape=[], dtype=Bool, value= False))
epoch time: 509161.697 ms, per step time: 758.810 ms
sh-4.3$
```

图 2.42 训练结果



打开 OBS 桶中 bert/model\_finetype 文件夹，可查看训练好的模型，即为训练成功



图 2.43 OBS 桶内微调模型

### 步骤 5 测试评价指标

若免费时长用尽，则需在 Notebook 列表界面点击“启动”后，重复步骤 2-3，再继续后续操作。

打开 code/src/config.py 文件

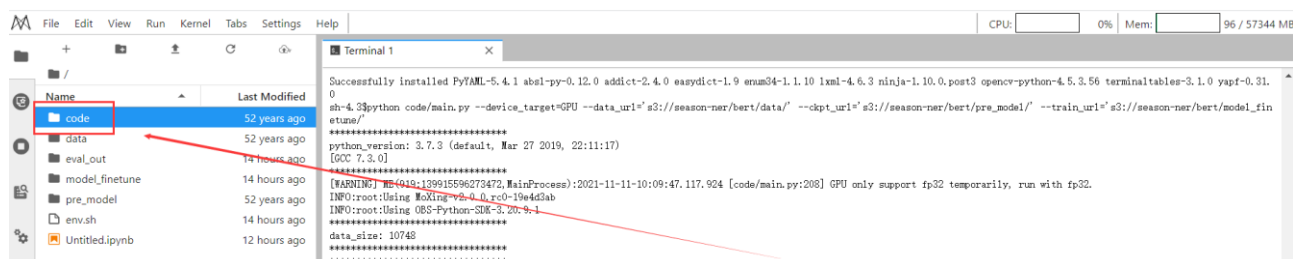


图 2.44 JupyterLab 内打开 code

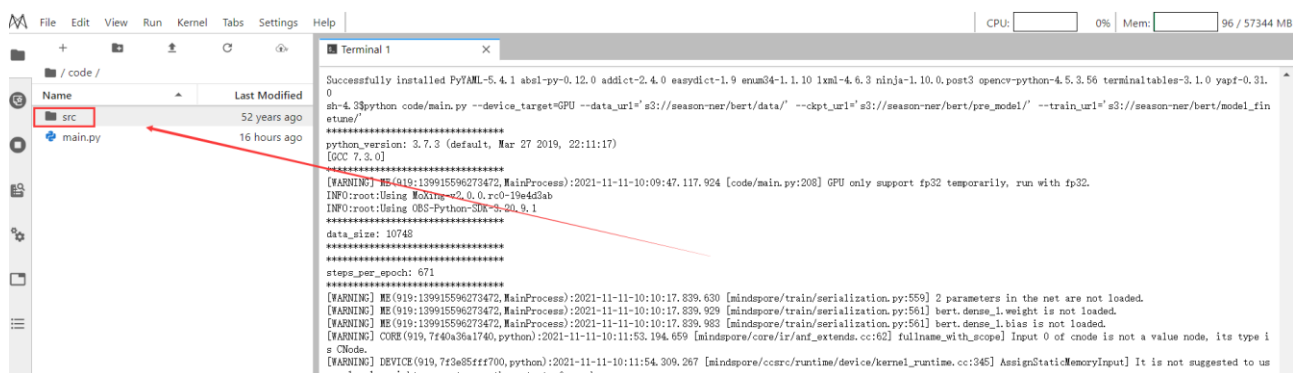


图 2.44 JupyterLab 内打开 code/src

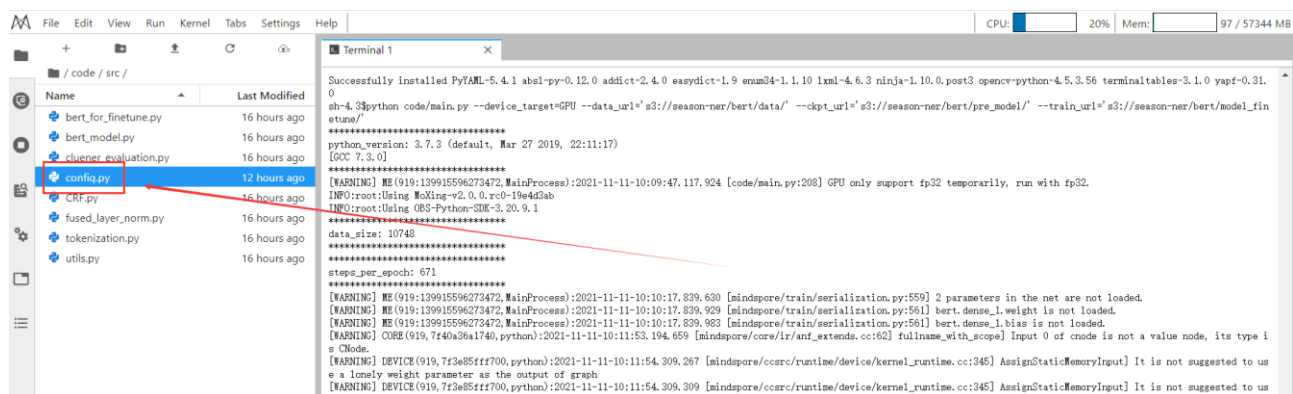


图 2.45 JupyterLab 内打开 code/src/config.py



修改第 24-47 行内容，主要将 “is\_train” 设为 False，并修改 “data\_file” 的内容，如下所示

```
24  cfg = edict({
25      'is_train': False, # 是否训练
26      'num_labels': 41, # 41, 类别数量
27      'schema_file': r'./data/schema.json', # 数据配置文件
28      'ckpt_prefix': 'bert-ner', # 无crf
29      # 'ckpt_prefix': 'bert-ner-crf', # 有crf
30      # 'data_file': r'./data/train.tf record', # 训练
31      'data_file': r'./data/dev.tf record', # 测试指标
32      # 'data_file': r'./data/dev.json', # 测试结果
33      'use_crf': False, # 是否使用crf
34
35      'epoch_num': 5, # 迭代次数
36      'batch_size': 16, # batch大小
37      'ckpt_dir': 'model_finetune', # 微调模型所在文件夹名
38      'pre_training_ckpt': './ckpt/bert_base.ckpt', # 预训练模型
39
40      'finetune_ckpt': './ckpt/bert-ner-5_671.ckpt', # 无crf
41      # 'finetune_ckpt': './ckpt/bert-ner-crf-5_671.ckpt', # 有crf
42      'label2id_file': './data/label2id.json', # label to id文件
43      'vocab_file': './data/vocab.txt', # 词表
44      'eval_out_file': 'ner_result.txt', # 无crf
45      # 'eval_out_file': 'ner_crf_result.txt', # 有crf
46      'optimizer': 'Lamb' # 优化器
47  })
```

图 2.46 修改后 config 文件

修改完，ctrl+s 保存

回到 Terminal 终端，输入以下命令

```
python code/main.py --device_target=GPU --data_url='s3://你的桶名/bert/data/' --ckpt_url='s3://你
的桶名/bert/model_finetune/' --train_url='s3://你的桶名/bert/eval_out/'
```

运行结果如下

```
sh-4.3$python code/main.py --device_target=GPU --data_url='s3://season-ner/bert/data/' --ckpt_url='s3://season-ner/bert/model_finetune/' --train_url='s3://season-ner/bert/eval_out/'
*****
python_version: 3.7.3 (default, Mar 27 2019, 22:11:17)
[GCC 7.3.0]
*****
[WARNING] ME(4134:140661161523008,MainProcess):2021-11-11-11:03:40.679.941 [code/main.py:208] GPU only support fp32 temporarily, run with fp32.
INFO:root:Using MoXing-v2.0.0.rc0-19e4d3ab
INFO:root:Using OBS-Python-SDK-3.20.9.1
*****
data_size: 1343
*****
Precision 0.915763
Recall 0.956560
F1 0.935717
*****
sh-4.3$
```

图 2.47 测试指标

### 步骤 6 启动微调训练（有 CRF）

建议重新启动 Notebook，即重新获取一小时时长，重复步骤 2-3 并进行后续操作，确保训练中途不会时长耗尽。

打开 code/src/config.py 文件，修改第 24-47 行内容，修改如下



```
24 cfg = edict({
25     'is_train': True, # 是否训练
26     'num_labels': 41, # 41, 类别数量
27     'schema_file': r'./data/schema.json', # 数据配置文件
28     # 'ckpt_prefix': 'bert-ner', # 无crf
29     'ckpt_prefix': 'bert-ner-crf', # 有crf
30     'data_file': r'./data/train.tf_record', # 训练
31     # 'data_file': r'./data/dev.tf_record', # 测试指标
32     # 'data_file': r'./data/dev.json', # 测试结果
33     'use_crf': True, # 是否使用crf
34
35     'epoch_num': 5, # 迭代次数
36     'batch_size': 16, # batch大小
37     'ckpt_dir': 'model_finetune', # 微调模型所在文件夹名
38     'pre_training_ckpt': './ckpt/bert_base.ckpt', # 预训练模型
39
40     # 'finetune_ckpt': './ckpt/bert-ner-5_671.ckpt', # 无crf
41     'finetune_ckpt': './ckpt/bert-ner-crf-5_671.ckpt', # 有crf
42     'label2id_file': './data/label2id.json', # label to id文件
43     'vocab_file': './data/vocab.txt', # 词表
44     # 'eval_out_file': 'ner_result.txt', # 无crf
45     'eval_out_file': 'ner_crf_result.txt', # 有crf
46     'optimizer': 'Lamb' # 优化器
47 })
```

图 2.48 修改后 config 文件

打开 Terminal 终端，输入以下命令（同步骤 4 中命令）

```
python code/main.py --device_target=GPU --data_url='s3://你的桶名/bert/data/' --ckpt_url='s3://你的桶名/bert/pre_model/' --train_url='s3://你的桶名/bert/model_finetune/'
```

同步骤 4，出现以下内容，则表明训练正常运行，继续耐心等待训练结束即可（约 40-50 分钟）。

```
epoch: 0, current epoch percent: 1.000, step: 671, outputs are (Tensor(shape=[], dtype=Float32, value= 21.332), Tensor(shape=[], dtype=Bool, value= False))
[WARNING] DEBUG (1396, 7f5f72174740, python):2021-11-11-15:30:54.040.978 [mindspore/ccsrc/debug/dump_proto.cc:467] ExportCNode] Operator must be a primitive
[WARNING] DEBUG (1396, 7f5f72174740, python):2021-11-11-15:30:54.045.255 [mindspore/ccsrc/debug/dump_proto.cc:231] SetValueToProto] Unsupported type FuncGraph
[WARNING] DEBUG (1396, 7f5f72174740, python):2021-11-11-15:30:54.045.271 [mindspore/ccsrc/debug/dump_proto.cc:231] SetValueToProto] Unsupported type FuncGraph
epoch time: 745156.383 ms, per step time: 1110.516 ms
```

图 2.49 epoch 信息

训练结束如下所示

```
epoch: 0, current epoch percent: 1.000, step: 671, outputs are (Tensor(shape=[], dtype=Float32, value= 21.332), Tensor(shape=[], dtype=Bool, value= False))
[WARNING] DEBUG (1396, 7f5f72174740, python):2021-11-11-15:30:54.040.978 [mindspore/ccsrc/debug/dump_proto.cc:467] ExportCNode] Operator must be a primitive
[WARNING] DEBUG (1396, 7f5f72174740, python):2021-11-11-15:30:54.045.255 [mindspore/ccsrc/debug/dump_proto.cc:231] SetValueToProto] Unsupported type FuncGraph
[WARNING] DEBUG (1396, 7f5f72174740, python):2021-11-11-15:30:54.045.271 [mindspore/ccsrc/debug/dump_proto.cc:231] SetValueToProto] Unsupported type FuncGraph
epoch time: 745156.383 ms, per step time: 1110.516 ms
epoch: 1, current epoch percent: 1.000, step: 1342, outputs are (Tensor(shape=[], dtype=Float32, value= 12.8181), Tensor(shape=[], dtype=Bool, value= False))
epoch time: 588978.733 ms, per step time: 877.763 ms
epoch: 2, current epoch percent: 1.000, step: 2013, outputs are (Tensor(shape=[], dtype=Float32, value= 9.49968), Tensor(shape=[], dtype=Bool, value= False))
epoch time: 587647.678 ms, per step time: 875.779 ms
epoch: 3, current epoch percent: 1.000, step: 2684, outputs are (Tensor(shape=[], dtype=Float32, value= 9.89726), Tensor(shape=[], dtype=Bool, value= False))
epoch time: 588452.508 ms, per step time: 876.978 ms
epoch: 4, current epoch percent: 1.000, step: 3355, outputs are (Tensor(shape=[], dtype=Float32, value= 7.0126), Tensor(shape=[], dtype=Bool, value= False))
epoch time: 587440.813 ms, per step time: 875.471 ms
sh-4.3$
```

图 2.50 训练结果

打开 OBS 桶，在 bert/model\_finetune 目录下可见训练好的模型，即训练成功。



图 2.51 OBS 桶内微调模型



## 步骤 7 测试指标

若时长终止，则需重新启动 Notebook，并重复步骤 2-3。

打开 code/src/config.py 文件，修改第 24-47 行，修改如下

```
24 cfg = edict({
25     'is_train': False, # 是否训练
26     'num_labels': 41, # 41, 类别数量
27     'schema_file': r'./data/schema.json', # 数据配置文件
28     # 'ckpt_prefix': 'bert-ner', # 无crf
29     'ckpt_prefix': 'bert-ner-crf', # 有crf
30     # 'data_file': r'./data/train.tf_record', # 训练
31     'data_file': r'./data/dev.tf_record', # 测试指标
32     # 'data_file': r'./data/dev.json', # 测试结果
33     'use_crf': True, # 是否使用crf
34
35     'epoch_num': 5, # 迭代次数
36     'batch_size': 16, # batch大小
37     'ckpt_dir': 'model_finetune', # 微调模型所在文件夹名
38     'pre_training_ckpt': './ckpt/bert_base.ckpt', # 预训练模型
39
40     # 'finetune_ckpt': './ckpt/bert-ner-5_671.ckpt', # 无crf
41     'finetune_ckpt': './ckpt/bert-ner-crf-5_671.ckpt', # 有crf
42     'label2id_file': './data/label2id.json', # label to id文件
43     'vocab_file': './data/vocab.txt', # 词表
44     # 'eval_out_file': 'ner_result.txt', # 无crf
45     'eval_out_file': 'ner_crf_result.txt', # 有crf
46     'optimizer': 'Lamb' # 优化器
47 })
```

图 2.52 修改后 config 文件

打开 Terminal 终端，输入以下命令，同步步骤 5 中命令相同

```
python code/main.py --device_target=GPU --data_url='s3://你的桶名/bert/data/' --ckpt_url='s3://你的桶名/bert/model_finetune/' --train_url='s3://你的桶名/bert/eval_out/'
```

若出现以下错误

```
Traceback (most recent call last):
  File "code/main.py", line 218, in <module>
    train()
  File "code/main.py", line 131, in train
    param_dict = load_checkpoint(cfg.pre_training_ckpt)
  File "/opt/conda/lib/python3.7/site-packages/mindspore/train/serialization.py", line 401, in load_checkpoint
    ckpt_file_name, filter_prefix = _check_checkpoint_param(ckpt_file_name, filter_prefix)
  File "/opt/conda/lib/python3.7/site-packages/mindspore/train/serialization.py", line 478, in _check_checkpoint_param
    raise ValueError("The checkpoint file does not exist.")
ValueError: The checkpoint file does not exist.
```

图 2.53 ValueError 错误

则为 config 文件尚未及时同步，返回 config 文件，ctrl+s 保存后，再次返回终端重试命令即可。

运行结果如下

```
sh-4.3$python code/main.py --device_target=GPU --data_url='s3://season-ner/bert/data/' --ckpt_url='s3://season-ner/bert/model_finetune/' --train_url='s3://season-ner/bert/eval_out/'
*****
python_version: 3.7.3 (default, Mar 27 2019, 22:11:17)
[GCC 7.3.0]
*****
[WARNING] ME (1798:140621325334336,MainProcess):2021-11-11-16:21:47.684.987 [code/main.py:208] GPU only support fp32 temporarily, run with fp32.
INFO:root:Using MoXing-v2.0.0.rc0-19e4d3ab
INFO:root:Using OBS-Python-SDK-3.20.9.1
*****
data_size: 1343
*****
*****
Precision 0.905928
Recall 0.957818
F1 0.931151
*****
sh-4.3$
```

图 2.54 测试指标



## 步骤 8 测试结果

打开 code/src/config.py 文件，修改第 24-47 行，修改如下

```
24 cfg = edict({
25     'is_train': False, # 是否训练
26     'num_labels': 41, # 41, 类别数量
27     'schema_file': r'./data/schema.json', # 数据配置文件
28     # 'ckpt_prefix': 'bert-ner', # 无crf
29     'ckpt_prefix': 'bert-ner-crf', # 有crf
30     # 'data_file': r'./data/train.tf_record', # 训练
31     # 'data_file': r'./data/dev.tf_record', # 测试指标
32     'data_file': r'./data/dev.json', # 测试结果
33     'use_crf': True, # 是否使用crf
34
35     'epoch_num': 5, # 迭代次数
36     'batch_size': 16, # batch大小
37     'ckpt_dir': 'model_finetune', # 微调模型所在文件夹名
38     'pre_training_ckpt': './ckpt/bert_base.ckpt', # 预训练模型
39
40     # 'finetune_ckpt': './ckpt/bert-ner-5_671.ckpt', # 无crf
41     'finetune_ckpt': './ckpt/bert-ner-crf-5_671.ckpt', # 有crf
42     'label2id_file': './data/label2id.json', # label to id文件
43     'vocab_file': './data/vocab.txt', # 词表
44     # 'eval_out_file': 'ner_result.txt', # 无crf
45     'eval_out_file': 'ner_crf_result.txt', # 有crf
46     'optimizer': 'Lamb' # 优化器
47 })
```

图 2.55 修改后 config

打开 Terminal 终端，输入以下命令，同步骤 5、7 相同

```
python code/main.py --device_target=GPU --data_url='s3://你的桶名/bert/data/' --ckpt_url='s3://你的桶名/bert/model_finetune/' --train_url='s3://你的桶名/bert/eval_out/'
```

运行部分结果如下

```
text 给学员们带去了丰富多彩的心理活动，使老师们不仅学习到了心理健康的知识，提升了自身的能力，
res: {'position': {'学员': [[1, 2]], '老师': [[18, 19]]}}
text 今年年初，坐在柏林国际电影节的大剧场里，林熙蕾第一次看到《文雀》全片。她才知道原来戏份还不算少，
res: {'address': {'柏林国际': [[7, 10]], 'name': {'林熙蕾': [[20, 22]], 'movie': {'《文雀》': [[28, 31]]}}}
text 在这个非常喜庆的日子里，我们首先掌声有请公园1872营销总监李杰为我们致辞。
res: {'address': {'公园': [[20, 21]], 'position': {'营销总监': [[26, 29]], 'name': {'李杰': [[30, 31]]}}}
text 姜哲中：公共之敌1-1》、《神机箭》、《7days》等多部热门影片的参加，
res: {'movie': {'姜哲中：公共之敌1-1》': [[0, 11]], '《神机箭》': [[13, 17]], '《7days》': [[19, 25]]}}
text 目前，日本松山海上保安部正在就此事进行调查。
res: {'government': {'日本松山海上保安部': [[3, 11]]}}
text 也就是说英国人在世博会上的英国馆，不会相办法表现出我是英国馆，从我的传统角度、文化角度，
res: {'scene': {'英国馆': [[27, 29]]}}
text 另外意大利的PlayGeneration杂志也刚刚给出了92%的高分。
res: {'address': {'意大利': [[2, 4]], 'book': {'PlayGeneration': [[6, 19]]}}}
sh-4.3$
```

图 2.56 测试结果

打开 OBS 桶，在 bert/eval\_out 目录下可查看生成的 ner\_crf\_result.txt 文件。



图 2.57 OBS 桶内 result 文件





## 2.4.2.2 训练作业实验

### 步骤 1 创建算法

点击左侧“算法管理”



图 2.58 ModelArts 左侧

点击上方的“创建”



图 2.59 算法管理

名称自定义，创建方式选择“自定义脚本”，AI 引擎选择“MPI”，代码目录选择 OBS 桶中的 bert/code 文件夹，启动文件选择 bert/code/main.py 文件，配置如下

★ 创建方式

自定义脚本 自定义镜像 ?

选择常用引擎创建训练作业。每个算法的代码目录最多支持1000个文件，文件深度不超过32，总大小不超过5GB。

★ AI引擎	MPI	mindspore_1.3.0-cuda_10.1-py_...	<input type="checkbox"/> 显示旧版引擎	<a href="#">引擎升级说明</a>
★ 代码目录	/season-ner/bert/code/		选择	?
★ 启动文件	/season-ner/bert/code/main.py		选择	?

图 2.60 自定义脚本

点击“添加输入数据配置”，映射名称自定义或者默认，代码路径参数填“data\_url”，添加约束选“是”，选择“数据存储位置”，如下

输入数据配置 为您的算法定义处理“输入数据”的参数，在您的算法代码中需要解析该参数获取到训练的数据集 ?

映射名称	数据来源1	代码路径参数	data_url	×
添加约束	<input checked="" type="radio"/> 是 <input type="radio"/> 否			
数据来源	<div>数据存储位置 ModelArts数据集</div>			

⊕ 添加输入数据配置

图 2.61 输入数据配置



点击“添加输出数据配置”，映射名称自定义或者默认，代码路径参数填“train\_url”，如下

输出数据配置 为您的算法定义处理“输出数据”的参数，在您的算法代码中需要解析该参数获取到训练的输出路径 (?)

映射名称	代码路径参数
输出数据1	train_url

⊕ 添加输出数据配置

图 2.62 输出数据配置

超参区域点击“增加超参”，名称填“device\_target”，类型选择“String”，默认值填“GPU”，如下

名称	类型	默认值	约束	操作
train_url	String		是	删除
device_target	String	GPU	是	删除

⊕ 增加超参

图 2.63 增加超参

添加训练约束选择“否”，点击下方“提交”

添加训练约束 ☐ 是 ☒ 否

配置费用: 免费 (?)

提交

图 2.64 点击提交

在“我的算法”里可以查看刚创建的算法

算法管理 使用指南

我的算法 我的订阅

创建

名称	标签	大小 (MB)	描述	创建时间	操作
ner-mind-gpu		0.11	--	2021/11/11 16:52:36 GMT+08:00	复制 删除 创建训练作业

图 2.65 算法管理

## 步骤 2 创建微调训练的训练作业（无 CRF）

启动 Notebook，将 bert/code/src/config.py 文件第 24-47 行修改如下（或者在本地修改后上传至 OBS 桶）

```
24 cfg = edict({
25     'is_train': True, # 是否训练
26     'num_labels': 41, # 41. 类别数量
27     'schema_file': './data/schema.json', # 数据配置文件
28     'ckpt_prefix': 'bert-ner', # 无crf
29     'ckpt_prefix': 'bert-ner-crf', # 有crf
30     'data_file': './data/train.tf_record', # 训练
31     'data_file': './data/dev.tf_record', # 测试指标
32     'data_file': './data/dev.json', # 测试结果
33     'use_crf': False, # 是否使用crf
34
35     'epoch_num': 5, # 迭代次数
36     'batch_size': 16, # batch大小
37     'ckpt_dir': 'model_finetune', # 微调模型所在文件夹名
38     'pre_training_ckpt': './ckpt/bert_base.ckpt', # 预训练模型
39
40     'finetune_ckpt': './ckpt/bert-ner-5.671.ckpt', # 无crf
41     'finetune_ckpt': './ckpt/bert-ner-crf-5.671.ckpt', # 有crf
42     'label2id_file': './data/label2id.json', # Label to id文件
43     'vocab_file': './data/vocab.txt', # 词表
44     'eval_out_file': 'ner_result.txt', # 无crf
45     'eval_out_file': 'ner_crf_result.txt', # 有crf
46     'optimizer': 'Lamb' # 优化器
47 })
```

图 2.66 修改后 config

进入 ModelArts，点击左侧“训练作业”





图 2.67 ModelArts 左侧

点击上方的“创建训练作业”



图 2.68 训练作业列表

名称自定义，算法选择刚创建的算法



图 2.69 选择算法

训练输入处，点击“选择数据存储位置”，选择 bert/data 文件夹，如下

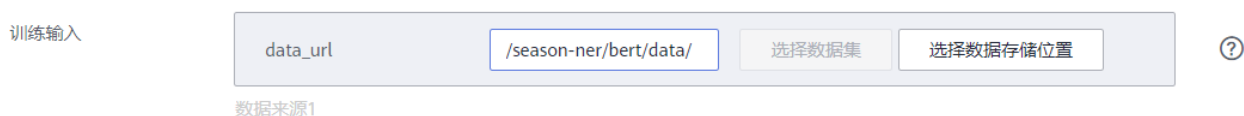


图 2.70 输入选择

训练输出处，点击“选择”，选择 bert/model\_finetune 文件夹，如下

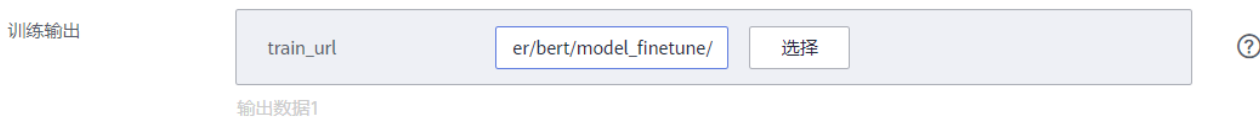


图 2.71 输出选择

超参处，点击“增加超参”，参数名填写“ckpt\_url”，参数值填写“s3://你的桶名/bert/pre\_model/”，如下



超参

device_target	=	GPU
data_url	=	/season-ner/bert/data/
train_url	=	/season-ner/bert/model_finetune/
ckpt_url	=	s3://season-ner/bert/pre_model/

+ 增加超参

图 2.72 增加超参

资源池选择“公共资源池”，资源类型选择“GPU”，规格选择“GPU: 1\*NVIDIA-V100(32GB)|CPU: 8核 64GB”，如下

\* 资源池 ☒ 公共资源池 ☐ 专属资源池

\* 资源类型 ☐ CPU ☒ GPU ☐ Ascend

\* 规格 GPU: 1\*NVIDIA-V100(32GB) | CPU: 8核 64GB

\* 计算节点个数  1

永久保存日志 ☐

日志30天后会被清理，打开按钮后可保存至指定OBS路径。您可以在作业详情页下载全部日志至本地。

图 2.73 资源选择

点击下方的“提交”即可

超参

device_target	=	GPU
data_url	=	/season-ner/bert/data/
train_url	=	/season-ner/bert/model_finetune/
ckpt_url	=	s3://season-ner/bert/pre_model/

+ 增加超参

环境变量 + 增加环境变量

\* 资源池 ☒ 公共资源池 ☐ 专属资源池

\* 资源类型 ☐ CPU ☒ GPU ☐ Ascend

\* 规格 GPU: 1\*NVIDIA-V100(32GB) | CPU: 8核 64GB

\* 计算节点个数  1

永久保存日志 ☐

日志30天后会被清理，打开按钮后可保存至指定OBS路径。您可以在作业详情页下载全部日志至本地。

配置费用 ¥28.00/小时

提交

图 2.74 点击提交

点击训练作业名称，即可进入日志查看界面

训练作业 <span>New</span>									
创建训练作业	只显示自己 <input checked="" type="checkbox"/>		高级搜索	全部状态	请输入名称查询				
名称	状态	运行时长 (min...)	创建时间	算法	规格	节点 (个数)	描述	创建人	操作
ner-mind-gp...	运行中	00:00:10	2021/11/13 22:55...	ner-mind-gpu	GPU: 1*NVIDIA-V100(32GB...	1	--	hw45811351	删除   重建   终止

图 2.75 训练作业列表

界面如下



训练作业 / ner-mind-gpu-no-crf

作业ID: 45c6894d-0253-4e72-8a3f-b0f27784a337

状态: 运行中

创建时间: 2021/11/13 22:55:36

运行时间: 00:00:45

算法名称: ner-mind-gpu

AI引擎: MPI | mindspore\_1.3.0-cuda\_10.1-py\_3.7-ubun...

代码目录: /season-ner/bert/code/

启动文件: /season-ner/bert/code/main.py

计算节点个数: 1

规格: GPU: 1\*NVIDIA-V100(32GB) | CPU: 8 核 64GB

资源占用情况

图 2.76 训练作业界面

出现如下信息，则表明训练正常，耐心等待训练结束即可（约为 30 分钟）

```
198 epoch: 0, current epoch percent: 1.000, step: 671, outputs are (Tensor(shape=
199 [], dtype=Float32, value= 0.0766297), Tensor(shape=[], dtype=Bool, value=
200 False))
201 [WARNING] DEBUG(160,7f9664202740,python):2021-11-14-14:16:09.450.883
202 [mindspore/ccsrc/debug/dump_proto.cc:467] ExportCNode] Operator must be a
203 primitive
204 [WARNING] DEBUG(160,7f9664202740,python):2021-11-14-14:16:09.452.264
205 [mindspore/ccsrc/debug/dump_proto.cc:231] SetValueToProto] Unsupported type
206 FuncGraph
207 [WARNING] DEBUG(160,7f9664202740,python):2021-11-14-14:16:09.452.283
208 [mindspore/ccsrc/debug/dump_proto.cc:231] SetValueToProto] Unsupported type
209 FuncGraph
210 epoch time: 413451.041 ms, per step time: 616.171 ms
```

图 2.77 epoch 信息

训练完成结果如下

训练作业 / ner-mind-gpu-no-crf

作业ID: f580452e-dfd8-4594-b921-ae273a2892f2

状态: 已完成

创建时间: 2021/11/14 14:08:21

运行时间: 00:27:51

算法名称: ner-mind-gpu

AI引擎: MPI | mindspore\_1.3.0-cuda\_10.1-py\_3.7-ubun...

代码目录: /season-ner/bert/code/

启动文件: /season-ner/bert/code/main.py

计算节点个数: 1

规格: GPU: 1\*NVIDIA-V100(32GB) | CPU: 8 核 64GB

资源占用情况

图 2.78 运行结果



## 步骤 3 创建测试指标的训练作业（无 CRF）

启动 Notebook，将 bert/code/src/config.py 文件第 24-47 行修改如下（或者在本地修改后上传至 OBS 桶）

```
24 cfg = edict({
25     'is_train': False, # 是否训练
26     'num_labels': 41, # 41, 类别数量
27     'schema_file': r'./data/schema.json', # 数据配置文件
28     'ckpt_prefix': 'bert-ner', # 无crf
29     # 'ckpt_prefix': 'bert-ner-crf', # 有crf
30     # 'data_file': r'./data/train.tf_record', # 训练
31     'data_file': r'./data/dev.tf_record', # 测试指标
32     # 'data_file': r'./data/dev.json', # 测试结果
33     'use_crf': False, # 是否使用crf
34
35     'epoch_num': 5, # 迭代次数
36     'batch_size': 16, # batch大小
37     'ckpt_dir': 'model_finetune', # 微调模型所在文件夹名
38     'pre_training_ckpt': './ckpt/bert_base.ckpt', # 预训练模型
39
40     'finetune_ckpt': './ckpt/bert-ner-5_671.ckpt', # 无crf
41     # 'finetune_ckpt': './ckpt/bert-ner-crf-5_671.ckpt', # 有crf
42     'label2id_file': './data/label2id.json', # label to id文件
43     'vocab_file': './data/vocab.txt', # 词表
44     'eval_out_file': 'ner_result.txt', # 无crf
45     # 'eval_out_file': 'ner_crf_result.txt', # 有crf
46     'optimizer': 'Lamb' # 优化器
47 })
```

图 2.79 修改后 config

创建训练作业，名称自定义，参数配置如下，其他配置同步步骤 2 中的微调训练的配置

训练输入	<div>data_url<div>/season-ner/bert/data/</div><div>选择数据集</div><div>选择数据存储位置</div></div> <div>数据来源1</div>			?
训练输出	<div>train_url<div>/season-ner/bert/eval_out/</div><div>选择</div></div> <div>输出数据1</div>			?
超参	<div>device_target<div>GPU</div></div> <div>data_url<div>/season-ner/bert/data/</div></div> <div>train_url<div>/season-ner/bert/eval_out/</div></div> <div>ckpt_url<div>s3://season-ner/bert/model_finetune/</div><div>增加超参</div></div>			
环境变量	<div>增加环境变量</div>			

图 2.80 参数配置

点击“提交”，运行结果如下



训练作业 / ner-mind-gpu-no-crf-eval

作业ID 3629c059-e343-414d-956e-5b68b508d390

状态 已完成

创建时间 2021/11/14 14:48:44

运行时间 00:02:16

算法名称 ner-mind-gpu

AI引擎 MPI | mindspore\_1.3.0-cuda\_10.1-py\_3.7-ubun...

代码目录 /season-ner/bert/code/

启动文件 /season-ner/bert/code/main.py

计算节点个数 1

规格 GPU: 1\*NVIDIA-V100(32GB) | CPU: 8 核 64GB

```
204 [ModelArts Service Log]exiting...
205 [ModelArts Service Log]exit with 0
206 [ModelArts Service Log][sidecar] training is completed
207 [ModelArts Service Log][sidecar] stop outputs_handler_pid = 61 by signal
SIGTERM
208 [ModelArts Service Log][INFO][2021/11/14 14:51:21,190]: output-handler
finalizing due to: [training finished]
209 [ModelArts Service Log][INFO][2021/11/14 14:51:21,190]: output-handler
finalized
210 [ModelArts Service Log][sidecar] stop toolkit_obs_sync_by_channels_pid = 77 by
signal SIGTERM
211 time="2021-11-14T14:51:21+08:00" level=info msg="the periodic upload task
exiting..." file="upload.go:59" Command=obs/sync_by_channels Component=ma-
training-toolkit Ctx=train_url Platform=ModelArts-Service
212 [ModelArts Service Log][sidecar] outputs_handler_pid = 61 ret_code is 0
213 [ModelArts Service Log][sidecar] toolkit_obs_sync_by_channels_pid = 77
ret_code is 0
214 [ModelArts Service Log][sidecar] upload_metrics_pid = 272
215 [ModelArts Service Log][sidecar] exiting at 2021-11-14-14:51:21
216 [ModelArts Service Log][sidecar] exit with 0
217 [ModelArts Service Log][sidecar] stop toolkit_obs_upload_pid = 33 by signal
SIGTERM
218 time="2021-11-14T14:51:21+08:00" level=info msg="the periodic upload task
exiting..." file="upload.go:59" Command=obs/upload Component=ma-training-
toolkit Platform=ModelArts-Service
219
```

资源占用情况

图 2.81 运行结果

滑动日志，可查看测试出的指标

```
186 -x NCCL_IB_DISABLE=0
187 -mca pm1 ob1 -mca btl ^openib
188 -mca plm_rsh_no_tree_spawn true /home/ma-user/anaconda/bin/python /home/ma-
user/modelarts/user-job-dir/code/main.py --device_target=GPU --
ckpt_url=s3://season-ner/bert/model_finetune/ --data_url=/home/ma-
user/modelarts/inputs/data_url_0/ --train_url=/home/ma-
user/modelarts/outputs/train_url_0/
189 *****
190 python_version: 3.7.10 (default, Jun 4 2021, 14:48:32)
191 [GCC 7.5.0]
192 *****
193 [WARNING] ME(160:139743894865728,MainProcess):2021-11-14-14:49:08.601.950
[/home/ma-user/modelarts/user-job-dir/code/main.py:208] GPU only support fp32
temporarily, run with fp32.
194 INFO:root:Using MoXing-v2.0.0.rc2.4b57a67b-4b57a67b
195 INFO:root:Using OBS-Python-SDK-3.20.9.1
196 *****
197 data_size: 1343
198 *****
199 =====
200 Precision 0.916560
201 Recall 0.952712
202 F1 0.934286
203 =====
204 [ModelArts Service Log]exiting...
205 [ModelArts Service Log]exit with 0
206 [ModelArts Service Log][sidecar] training is completed
207 [ModelArts Service Log][sidecar] stop outputs_handler_pid = 61 by signal
```

资源占用情况

图 2.82 测试指标



## 步骤 4 创建微调训练的训练作业（有 CRF）

启动 Notebook，将 bert/code/src/config.py 文件第 24-47 行修改如下（或者在本地修改后上传至 OBS 桶）

```
24 cfg = edict({
25     'is_train': True, # 是否训练
26     'num_labels': 41, # 41, 类别数量
27     'schema_file': r'./data/schema.json', # 数据配置文件
28     # 'ckpt_prefix': 'bert-ner', # 无crf
29     'ckpt_prefix': 'bert-ner-crf', # 有crf
30     'data_file': r'./data/train.tf_record', # 训练
31     # 'data_file': r'./data/dev.tf_record', # 测试指标
32     # 'data_file': r'./data/dev.json', # 测试结果
33     'use_crf': True, # 是否使用crf
34
35     'epoch_num': 5, # 迭代次数
36     'batch_size': 16, # batch大小
37     'ckpt_dir': 'model_finetune', # 微调模型所在文件夹名
38     'pre_training_ckpt': './ckpt/bert_base.ckpt', # 预训练模型
39
40     # 'finetune_ckpt': './ckpt/bert-ner-5_671.ckpt', # 无crf
41     'finetune_ckpt': './ckpt/bert-ner-crf-5_671.ckpt', # 有crf
42     'label2id_file': './data/label2id.json', # label to id文件
43     'vocab_file': './data/vocab.txt', # 词表
44     # 'eval_out_file': 'ner_result.txt', # 无crf
45     'eval_out_file': 'ner_crf_result.txt', # 有crf
46     'optimizer': 'Lamb' # 优化器
47 })
```

图 2.83 修改后 config

创建训练作业，同步骤 2

或者更快捷的方式，进入训练作业列表界面，点击步骤 2 运行结束的训练作业后的“重建”按钮

训练作业 <span>New</span>									
<div>创建训练作业</div> <div>只显示自己 <input checked="" type="checkbox"/> 高级搜索 全部状态 请输入名称查询</div>									
名称	状态	运行时长 (h:m:s)	创建时间	算法	规格	节点 (个数)	描述	创建人	操作
ner-mind-gp...	已完成	00:02:16	2021/11/14 14:48...	ner-mind-gpu	GPU: 1*NVIDIA-V100(32GB...	1	--	hw45811351	删除 重建 终止
ner-mind-gp...	已完成	00:27:51	2021/11/14 14:08...	ner-mind-gpu	GPU: 1*NVIDIA-V100(32GB...	1	--	hw45811351	删除 重建 终止

图 2.84 重建训练作业

修改名称，要求不重名

创建训练作业

[返回训练作业列表](#)

评价 使用指南 建议反馈

\* 名称

ner-mind-gpu-crf

描述

0/256

\* 算法

我的算法 我的订阅

创建

ner-mind-gpu

名称	AI 引擎	标签	大小 (MB)	描述	创建时间
----	-------	----	---------	----	------

图 2.85 修改名称



点击“提交”即可，出现以下信息，则表明训练正常运行

```
[mindspore/train/serialization.py:561] loss.transitions is not loaded.
197 [WARNING] CORE(160,7f56a38e5740,python):2021-11-14-15:01:55.497.851
[mindspore/core/ir/anf_extends.cc:62] fullname_with_scope] Input 0 of cnode is
not a value node, its type is CNode.
198 [WARNING] DEVICE(160,7f53dd7fe700,python):2021-11-14-15:01:58.094.892
[mindspore/ccsrc/runtime/device/kernel_runtime.cc:345]
AssignStaticMemoryInput] It is not suggested to use a lonely weight parameter
as the output of graph
199 [WARNING] DEVICE(160,7f53dd7fe700,python):2021-11-14-15:01:58.094.916
[mindspore/ccsrc/runtime/device/kernel_runtime.cc:345]
AssignStaticMemoryInput] It is not suggested to use a lonely weight parameter
as the output of graph
200 epoch: 0, current epoch percent: 1.000, step: 671, outputs are (Tensor(shape=
[], dtype=Float32, value= 24.3396), Tensor(shape=[], dtype=Bool, value=
False))
201 [WARNING] DEBUG(160,7f56a38e5740,python):2021-11-14-15:08:19.120.621
[mindspore/ccsrc/debug/dump_proto.cc:467] ExportCNode] Operator must be a
primitive
202 [WARNING] DEBUG(160,7f56a38e5740,python):2021-11-14-15:08:19.125.122
[mindspore/ccsrc/debug/dump_proto.cc:231] SetValueToProto] Unsupported type
FuncGraph
203 [WARNING] DEBUG(160,7f56a38e5740,python):2021-11-14-15:08:19.125.140
[mindspore/ccsrc/debug/dump_proto.cc:231] SetValueToProto] Unsupported type
FuncGraph
204 epoch time: 515411.694 ms, per step time: 768.125 ms
205
```

图 2.86 epoch 信息

训练完成如下

训练作业 / ner-mind-gpu-cr

作业ID 97e29cec-6695-4697-967f-59d46cd8e32e

状态 已完成

创建时间 2021/11/14 14:58:49

运行时间 00:34:30

算法名称 ner-mind-gpu

AI引擎 MPI | mindspore\_1.3.0-cuda\_10.1-py\_3.7-ubun...

代码目录 /season-ner/bert/code/

启动文件 /season-ner/bert/code/main.py

计算节点个数 1

规格 GPU: 1\*NVIDIA-V100(32GB) | CPU: 8 核 64GB

```
223 [ModelArts Service Log]exiting...
224 [ModelArts Service Log]exit with 0
225 [ModelArts Service Log][sidecar] training is completed
226 [ModelArts Service Log][sidecar] stop outputs_handler_pid = 61 by signal
SIGTERM
227 [ModelArts Service Log][INFO][2021/11/14 15:33:37,363]: output-handler
finalizing due to: [training finished]
228 [ModelArts Service Log][INFO][2021/11/14 15:33:37,363]: output-handler
finalized
229 [ModelArts Service Log][sidecar] stop toolkit_obs_sync_by_channels_pid = 76 by
signal SIGTERM
230 time="2021-11-14T15:33:39+08:00" level=info msg="the periodic upload task
exiting..." file="upload.go:59" Command=obs/sync_by_channels Component=ma-
training-toolkit Ctx=train_url Platform=ModelArts-Service
231 [ModelArts Service Log][sidecar] outputs_handler_pid = 61 ret_code is 0
232 [ModelArts Service Log][sidecar] toolkit_obs_sync_by_channels_pid = 76
ret_code is 0
233 [ModelArts Service Log][sidecar] upload_metrics_pid = 2203
234 [ModelArts Service Log][sidecar] exiting at 2021-11-14-15:33:41
235 [ModelArts Service Log][sidecar] exit with 0
236 [ModelArts Service Log][sidecar] stop toolkit_obs_upload_pid = 33 by signal
SIGTERM
237 time="2021-11-14T15:33:41+08:00" level=info msg="the periodic upload task
exiting..." file="upload.go:59" Command=obs/upload Component=ma-training-
toolkit Platform=ModelArts-Service
238
```

资源占用情况

图 2.87 运行结果



## 步骤 5 创建测试指标的训练作业（有 CRF）

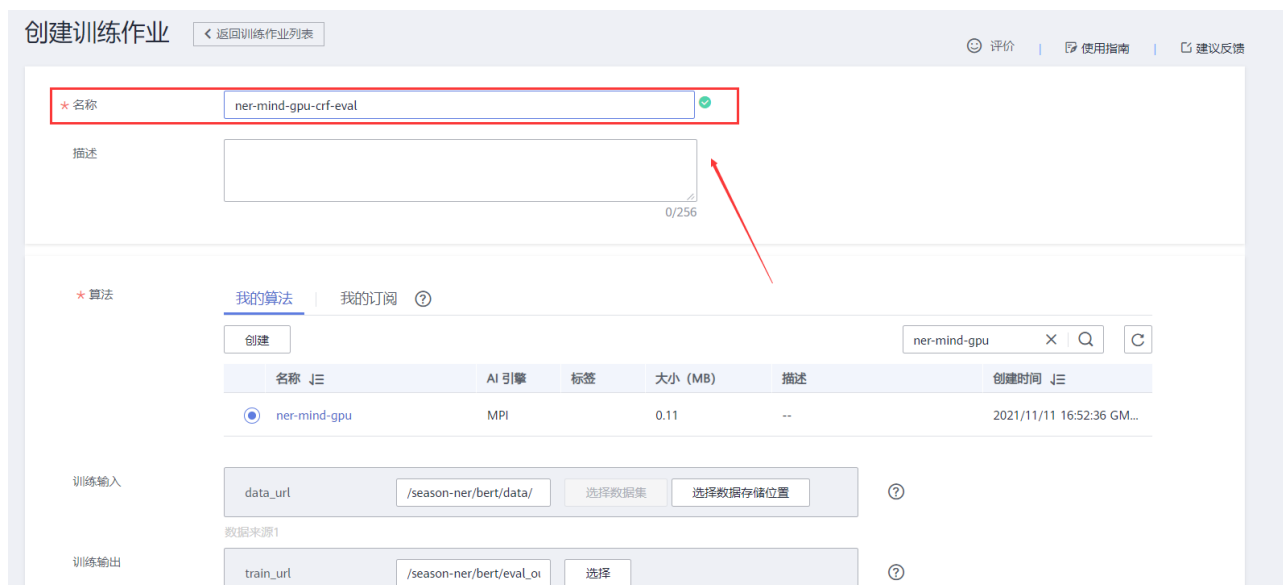
启动 Notebook，将 bert/code/src/config.py 文件第 24-47 行修改如下（或者在本地修改后上传至 OBS 桶）

```
24 cfg = edict({
25     'is_train': False, # 是否训练
26     'num_labels': 41, # 41, 类别数量
27     'schema_file': r'./data/schema.json', # 数据配置文件
28     # 'ckpt_prefix': 'bert-ner', # 无crf
29     'ckpt_prefix': 'bert-ner-crf', # 有crf
30     # 'data_file': r'./data/train.tf_record', # 训练
31     'data_file': r'./data/dev.tf_record', # 测试指标
32     # 'data_file': r'./data/dev.json', # 测试结果
33     'use_crf': True, # 是否使用crf
34
35     'epoch_num': 5, # 迭代次数
36     'batch_size': 16, # batch大小
37     'ckpt_dir': 'model_finetune', # 微调模型所在文件夹名
38     'pre_training_ckpt': './ckpt/bert_base.ckpt', # 预训练模型
39
40     # 'finetune_ckpt': './ckpt/bert-ner-5_671.ckpt', # 无crf
41     'finetune_ckpt': './ckpt/bert-ner-crf-5_671.ckpt', # 有crf
42     'label2id_file': './data/label2id.json', # label to id文件
43     'vocab_file': './data/vocab.txt', # 词表
44     # 'eval_out_file': 'ner_result.txt', # 无crf
45     'eval_out_file': 'ner_crf_result.txt', # 有crf
46     'optimizer': 'Lamb' # 优化器
47 })
```

图 2.88 修改后 config

创建训练作业，同步步骤 3

或者更快捷的方式，进入训练作业列表界面，点击步骤 3 运行结束的训练作业后的“重建”按钮修改名称，要求不重名



创建训练作业

返回训练作业列表

评价 | 使用指南 | 建议反馈

\* 名称: ner-mind-gpu-crf-eval

描述: 0/256

\* 算法: 我的算法 | 我的订阅

创建: ner-mind-gpu x | Q | C

名称	AI 引擎	标签	大小 (MB)	描述	创建时间
ner-mind-gpu	MPI		0.11	--	2021/11/11 16:52:36 GM...

训练输入: data\_url: /season-ner/bert/data/ 选择数据集 选择数据存储位置

训练输出: train\_url: /season-ner/bert/eval\_o/ 选择

图 2.89 修改名称

点击“提交”，运行结果如下





训练作业 / ner-mind-gpu-crf-eval

重建 创建模型 终止

作业ID: 2f996020-46c0-4e85-9db7-83fcc8759cf0

状态: 已完成

创建时间: 2021/11/14 15:40:04

运行时间: 00:03:15

算法名称: ner-mind-gpu

AI引擎: MPI | mindspore\_1.3.0-cuda\_10.1-py\_3.7-ubun...

代码目录: /season-ner/bert/code/

启动文件: /season-ner/bert/code/main.py

计算节点个数: 1

规格: GPU: 1\*NVIDIA-V100(32GB) | CPU: 8 核 64GB

```
204 [ModelArts Service Log]exiting...
205 [ModelArts Service Log]exit with 0
206 [ModelArts Service Log][sidecar] training is completed
207 [ModelArts Service Log][sidecar] stop outputs_handler_pid = 61 by signal
SIGTERM
208 [ModelArts Service Log][INFO][2021/11/14 15:43:42,754]: output-handler
finalizing due to: [training finished]
209 [ModelArts Service Log][INFO][2021/11/14 15:43:42,754]: output-handler
finalized
210 [ModelArts Service Log][sidecar] stop toolkit_obs_sync_by_channels_pid = 76 by
signal SIGTERM
211 time="2021-11-14T15:43:42+08:00" level=info msg="the periodic upload task
exiting..." file="upload.go:59" Command=obs/sync_by_channels Component=ma-
training-toolkit Ctx=train_url Platform=ModelArts-Service
212 [ModelArts Service Log][sidecar] outputs_handler_pid = 61 ret_code is 0
213 [ModelArts Service Log][sidecar] toolkit_obs_sync_by_channels_pid = 76
ret_code is 0
214 [ModelArts Service Log][sidecar] upload_metrics_pid = 331
215 [ModelArts Service Log][sidecar] exiting at 2021-11-14-15:43:43
216 [ModelArts Service Log][sidecar] exit with 0
217 [ModelArts Service Log][sidecar] stop toolkit_obs_upload_pid = 34 by signal
SIGTERM
218 time="2021-11-14T15:43:43+08:00" level=info msg="the periodic upload task
exiting..." file="upload.go:59" Command=obs/upload Component=ma-training-
toolkit Platform=ModelArts-Service
219
```

资源占用情况

图 2.90 运行结果

滑动日志，可查看测试出的指标

```
user/modelarts/outputs/train_url_0/
189 *****
190 python_version: 3.7.10 (default, Jun 4 2021, 14:48:32)
191 [GCC 7.5.0]
192 *****
193 [WARNING] ME(161:139881450604352,MainProcess):2021-11-14-15:40:31.952.80
[/home/ma-user/modelarts/user-job-dir/code/main.py:208] GPU only support fp32
temporarily, run with fp32.
194 INFO:root:Using MoXing-v2.0.0.rc2.4b57a67b-4b57a67b
195 INFO:root:Using OBS-Python-SDK-3.20.9.1
196 *****
197 data_size: 1343
198 *****
199 =====
200 Precision 0.907249
201 Recall 0.957670
202 F1 0.931778
203 =====
204 [ModelArts Service Log]exiting...
205 [ModelArts Service Log]exit with 0
206 [ModelArts Service Log][sidecar] training is completed
207 [ModelArts Service Log][sidecar] stop outputs_handler_pid = 61 by signal
SIGTERM
208 [ModelArts Service Log][INFO][2021/11/14 15:43:42,754]: output-handler
finalizing due to: [training finished]
209 [ModelArts Service Log][INFO][2021/11/14 15:43:42,754]: output-handler
```

资源占用情况

图 2.91 测试指标

步骤 6 创建测试结果的训练作业（有 CRF）



启动 Notebook, 将 bert/code/src/config.py 文件第 24-47 行修改如下(或者在本地修改后上传至 OBS 桶)

```
24 cfg = edict({
25     'is_train': False, # 是否训练
26     'num_labels': 41, # 41, 类别数量
27     'schema_file': r'./data/schema.json', # 数据配置文件
28     # 'ckpt_prefix': 'bert-ner', # 无crf
29     'ckpt_prefix': 'bert-ner-crf', # 有crf
30     # 'data_file': r'./data/train.tf_record', # 训练
31     # 'data_file': r'./data/dev.tf_record', # 测试指标
32     'data_file': r'./data/dev.json', # 测试结果
33     'use_crf': True, # 是否使用crf
34
35     'epoch_num': 5, # 迭代次数
36     'batch_size': 16, # batch大小
37     'ckpt_dir': 'model_finetune', # 微调模型所在文件夹名
38     'pre_training_ckpt': './ckpt/bert_base.ckpt', # 预训练模型
39
40     # 'finetune_ckpt': './ckpt/bert-ner-5_671.ckpt', # 无crf
41     'finetune_ckpt': './ckpt/bert-ner-crf-5_671.ckpt', # 有crf
42     'label2id_file': './data/label2id.json', # label to id文件
43     'vocab_file': './data/vocab.txt', # 词表
44     # 'eval_out_file': 'ner_result.txt', # 无crf
45     'eval_out_file': 'ner_crf_result.txt', # 有crf
46     'optimizer': 'Lamb' # 优化器
47 })
```

图 2.92 修改后 config

创建训练作业, 同步骤 3 和步骤 5

或者更快捷的方式, 进入训练作业列表界面, 点击步骤 3 或步骤 5 运行结束的训练作业后的“重建”按钮

修改名称, 要求不重名

创建训练作业

名称: ner-mind-gpu-crf-eval-out

描述: 0/256

算法: 我的算法 | 我的订阅

创建: ner-mind-gpu

名称	AI 引擎	标签	大小 (MB)	描述	创建时间
ner-mind-gpu	MPI		0.11	--	2021/11/11 16:52:36 GM...

训练输入: data\_url: /season-ner/bert/data/

训练输出: 数据源1

图 2.93 修改名称

点击“提交”, 运行结果如下



训练作业 / ner-mind-gpu-crf-eval-out

重建 | 创建模型 | 终止 | 删除

作业ID	b2d52f15-bba2-496c-a28e-a561f9e4811f
状态	已完成
创建时间	2021/11/14 15:46:44
运行时间	00:04:00

---

算法名称	ner-mind-gpu
AI引擎	MPI   mindspore_1.3.0-cuda_10.1-py_3.7-ubun...
代码目录	/season-ner/bert/code/
启动文件	/season-ner/bert/code/main.py

---

计算节点个数	1
规格	GPU: 1*NVIDIA-V100(32GB)   CPU: 8 核 64GB

2883 [ModelArts Service Log]exiting...  
2884 [ModelArts Service Log]exit with 0  
2885 [ModelArts Service Log][sidecar] training is completed  
2886 [ModelArts Service Log][sidecar] stop outputs\_handler\_pid = 62 by signal SIGTERM  
2887 [ModelArts Service Log][INFO][2021/11/14 15:51:07,426]: output-handler finalizing due to: [training finished]  
2888 [ModelArts Service Log][INFO][2021/11/14 15:51:07,426]: output-handler finalized  
2889 [ModelArts Service Log][sidecar] stop toolkit\_obs\_sync\_by\_channels\_pid = 77 by signal SIGTERM  
2890 time="2021-11-14T15:51:08+08:00" level=info msg="the periodic upload task exiting..." file="upload.go:59" Command=obs/sync\_by\_channels Component=ma-training-toolkit Ctx=train\_url Platform=ModelArts-Service  
2891 [ModelArts Service Log][sidecar] outputs\_handler\_pid = 62 ret\_code is 0  
2892 [ModelArts Service Log][sidecar] toolkit\_obs\_sync\_by\_channels\_pid = 77 ret\_code is 0  
2893 [ModelArts Service Log][sidecar] upload\_metrics\_pid = 376  
2894 [ModelArts Service Log][sidecar] exiting at 2021-11-14-15:51:08  
2895 [ModelArts Service Log][sidecar] exit with 0  
2896 [ModelArts Service Log][sidecar] stop toolkit\_obs\_upload\_pid = 34 by signal SIGTERM  
2897 time="2021-11-14T15:51:08+08:00" level=info msg="the periodic upload task exiting..." file="upload.go:59" Command=obs/upload Component=ma-training-toolkit Platform=ModelArts-Service  
2898

资源占用情况

图 2.94 运行结果

滑动日志，可查看测试显示的结果

2864 res: {'position': {'记者': [[13, 14]], '负责人': [[22, 24]]}}  
2865 text 美女收藏家创下中国当代油画新的拍卖天价纪录。  
2866 res: {'position': {'收藏家': [[2, 4]]}}  
2867 text 金管局于2008年1月10日已经将调查结果转交给证监会。  
2868 res: {'government': {'金管局': [[0, 2]], '证监会': [[24, 26]]}}  
2869 text 给学员们带去了丰富多彩的心理活动，使老师们不仅学习到了心理健康的知识，提升了自身的能力，  
2870 res: {'position': {'学员': [[1, 2]], '老师': [[18, 19]]}}  
2871 text 今年年初，坐在柏林国际电影节的大剧场里，林熙蕾第一次看到《文雀》全片。她才知道原来戏份还不算少，  
2872 res: {'address': {'柏林': [[7, 8]]}, 'name': {'林熙蕾': [[20, 22]]}, 'movie': {'《文雀》': [[28, 31]]}}  
2873 text 在这个非常喜庆的日子里，我们首先掌声有请公园1872营销总监李杰为我们致辞。  
2874 res: {'address': {'公园': [[20, 21]]}, 'position': {'营销总监': [[26, 29]]}, 'name': {'李杰': [[30, 31]]}}  
2875 text 姜哲中：公共之敌1-1》、《神机箭》、《7days》等多部热门影片的参加，  
2876 res: {'movie': {'姜哲中：公共之敌1-1': [[0, 11]], '《神机箭》': [[13, 17]], '《7days》': [[19, 25]]}}  
2877 text 目前，日本松山海上保安部正在就此事进行调查。  
2878 res: {'government': {'日本松山海上保安部': [[3, 11]]}}  
2879 text 也就是说英国人在世博会上的英国馆，不会相办法表现出我是英国馆，从我的传统角度、文化角度，  
2880 res: {'scene': {'英国馆': [[27, 29]]}}  
2881 text 另外意大利的PlayGeneration杂志也刚刚给出了92%的高分。  
2882 res: {'address': {'意': [[2, 2]]}, 'book': {'PlayGeneration': [[6, 19]]}}  
2883 [ModelArts Service Log]exiting...  
2884 [ModelArts Service Log]exit with 0  
2885 [ModelArts Service Log][sidecar] training is completed

资源占用情况

图 2.95 测试结果

打开 OBS 桶，可查看测试出的 result 文件



图 2.96 OBS 桶内 result 文件

## 2.5 实验总结

本章实验在华为云 ModelArts 平台上使用 MindSpore 完整地实现了一个基于 BERT+CRF 的命名实体识别模型，BERT+CRF 是当前性能最好的序列标注架构之一，通过实验使学员了解最前沿的算法模型，也进一步加深对命名实体识别，BERT 等相关理论的理解。可以在该实验的基础上进一步开发其他类型的实体识别模型。