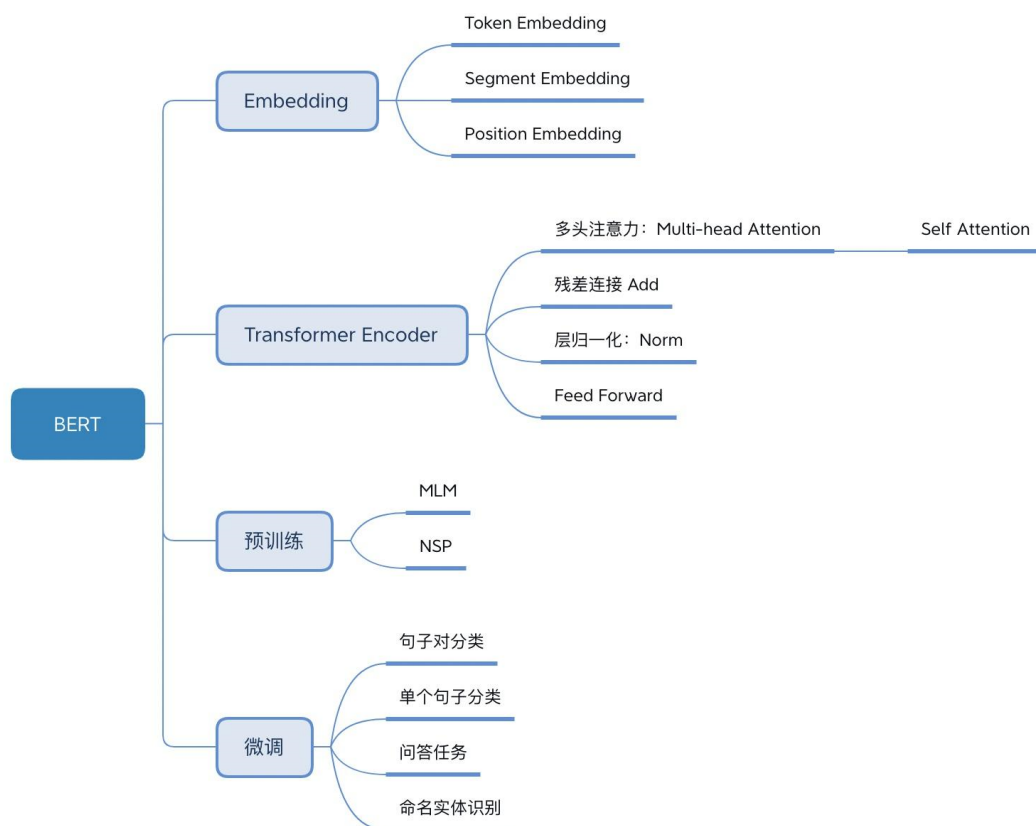


## 一、BERT 简介



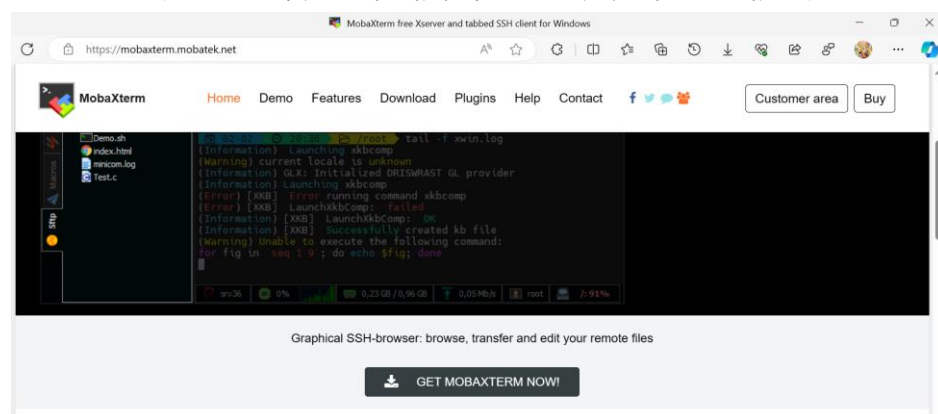
## 二、BERT 环境配置

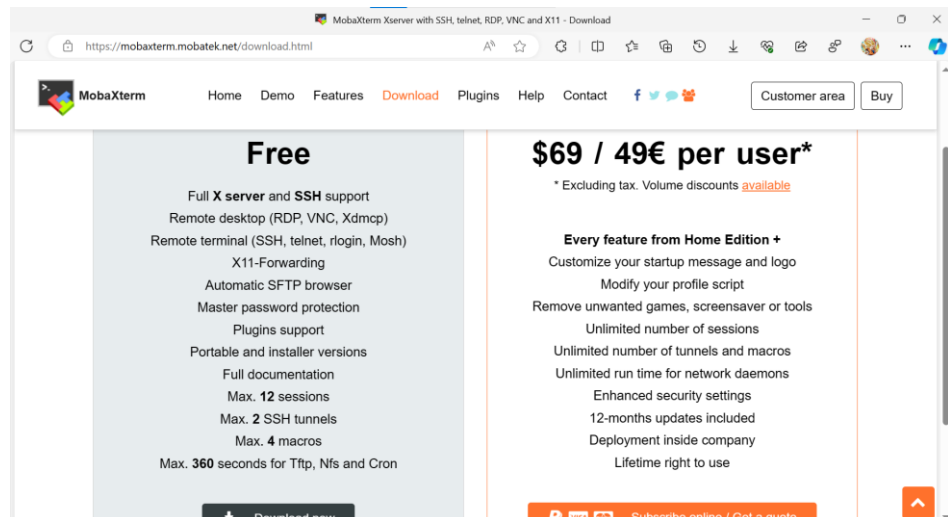
### 1、远程服务器部署

我的电脑跑不动，只能用远程服务器跑，上世纪白嫖的实验室服务器就排上了用场

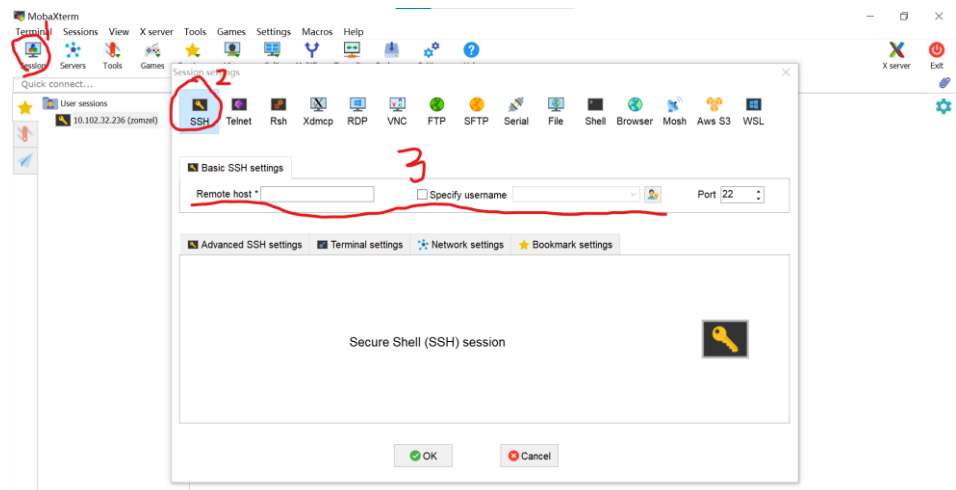
#### (1) MobaXterm

mobaXterm 用于远程服务器的连接，官网下载免费的就行直接可用





## 安装运行 mobaxTerm



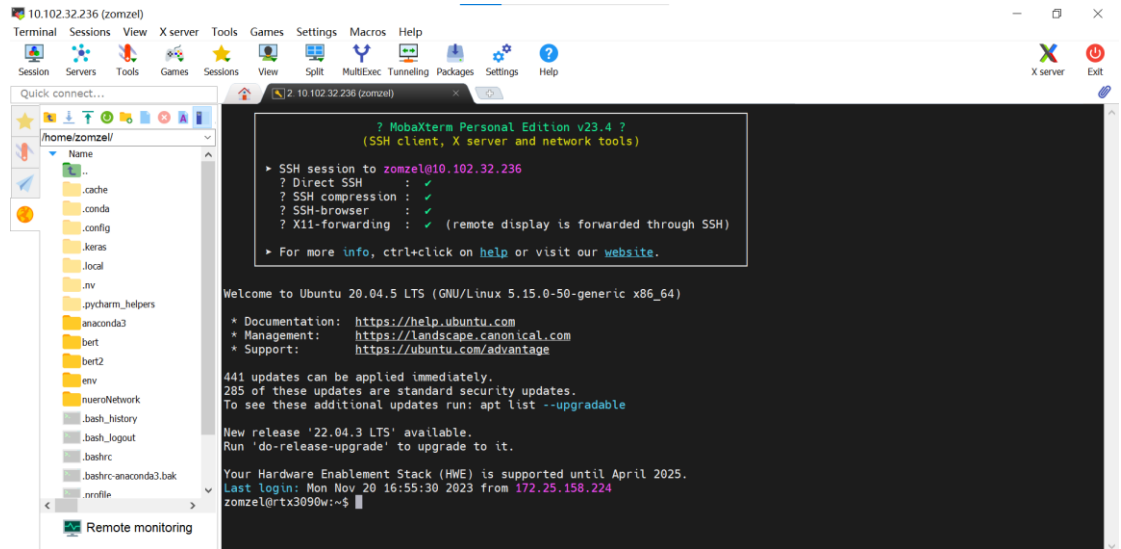
点击 session 创建会话，然后通过 ssh 进行连接，输入主机名和用户名进行连接（可能会要输入密码，需要跑 BERT 的小伙伴这里提供我的服务器，我已经在远程服务器上创建好了虚拟环境和运行配置）

username: zomzel

password: g9J&H^17e8nH

ip: 10.102.32.236

连接成功如下图所示。



使用 conda 创建运行 bert 的虚拟环境:

```
conda create -n bert_3_7 python=3.7
```

激活虚拟环境:

```
source activate bert_3_7
```

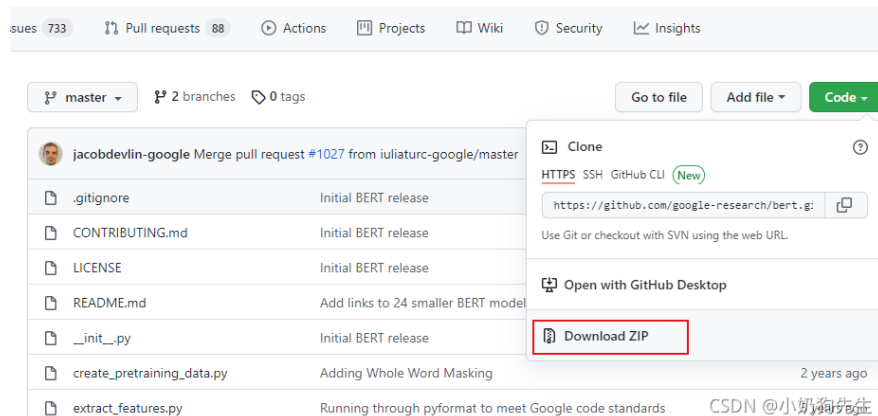
(运行需要的包不急下)

创建一个文件夹用于与本地项目文件进行映射

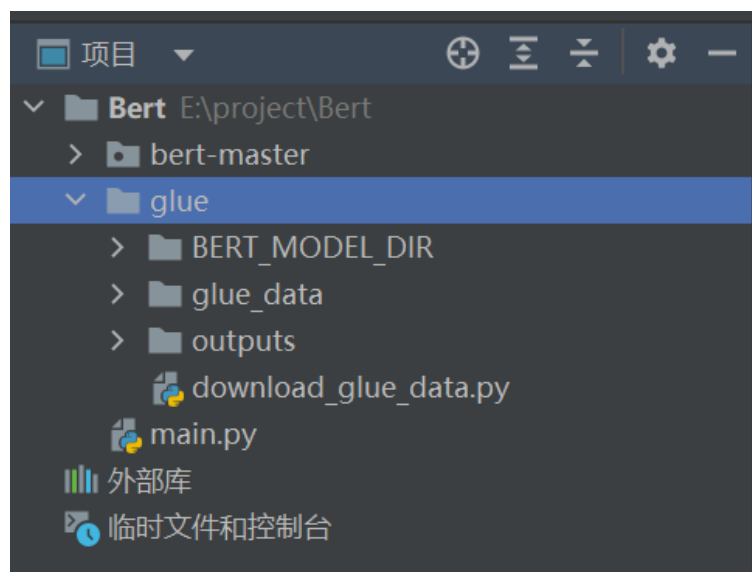
```
mkdir bert
```

## 2、BERT 项目代码和模型 git 到本地，在 pycharm 中打开

### (1) 下载 bert 源码



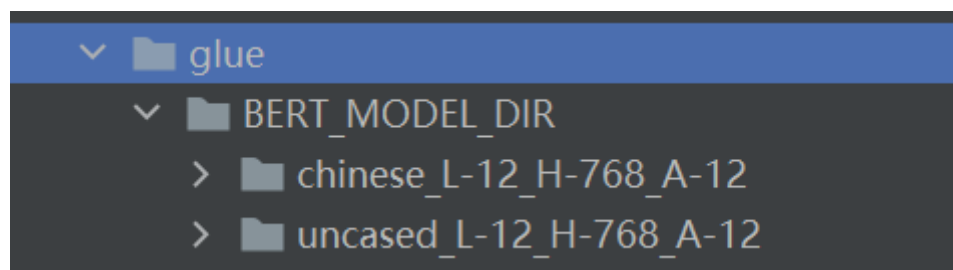
- (2) 本地新建一个文件夹 bert 用于存放项目文件，将压缩包 bert-master（这是源码）解压到这个文件夹中，在 bert 文件夹中新建一个文件夹 glue 用于存放模型文件和训练测试数据，在 glue 文件夹中新建 BERT\_MODEL\_DIR 用于存放模型文件，新建文件夹 glue\_data 用于存放训练语料库数据，outputs 文件夹用于模型输出。最后的结构如下图所示：



(3) BERT 的 github 主页下载预训练模型

- BERT-Large, Uncased (Whole Word Masking) : 24-layer, 1024-hidden, 16-heads, 340M
- BERT-Large, Cased (Whole Word Masking) : 24-layer, 1024-hidden, 16-heads, 340M pa
- BERT-Base, Uncased : 12-layer, 768-hidden, 12-heads, 110M parameters
- BERT-Large, Uncased : 24-layer, 1024-hidden, 16-heads, 340M parameters
- BERT-Base, Cased : 12-layer, 768-hidden, 12-heads , 110M parameters
- BERT-Large, Cased : 24-layer, 1024-hidden, 16-heads, 340M parameters
- BERT-Base, Multilingual Cased (New, recommended) : 104 languages, 12-layer, 768-hi  
parameters
- BERT-Base, Multilingual Uncased (Orig, not recommended) (Not recommended, use  
instead): 102 languages, 12-layer, 768-hidden, 12-heads, 110M parameters
- BERT-Base, Chinese : Chinese Simplified and Traditional, 12-layer, 768-hidden, 12-he

下载 Base 基础的预训练模型就行，下载后将压缩包解压到刚刚建立的 BERT\_MODEL\_DIR 文件夹中，如下图所示（这里多下载了一个 chinese 的不需要管他）：



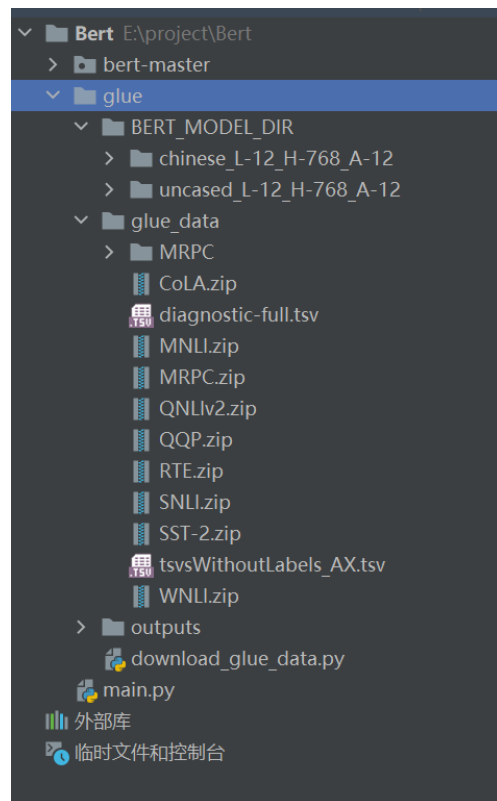
(4) 下载语料库数据

本次实验使用 MRPC 语料库进行句子的分类任务，需要先下载 GLUE 数据集。GLUE 是一个统称，里面有很多个独立的数据集，例如如果只使用 MRPC 进行测试，那可以只下载 MRPC 数据集，如果出现无法下载的情况，可以使用我下载好的：

链接: <https://pan.baidu.com/s/1rhy138f0wPNgTZwjO0CV4g>

提取码: dkf6

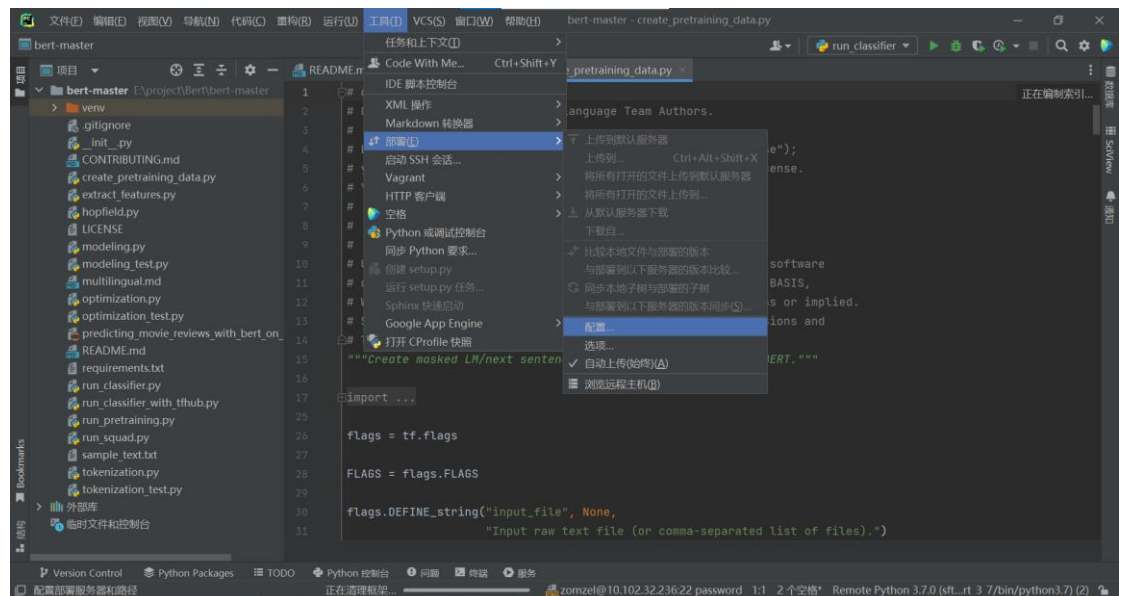
将 glue\_data 解压到 glue 文件夹中 (好像之前创建 glue\_data 文件夹多余了) 然后将 glue\_data 中的 MRPC 压缩包解压到当前文件夹。至此所有文件准备完毕,如下图所示:



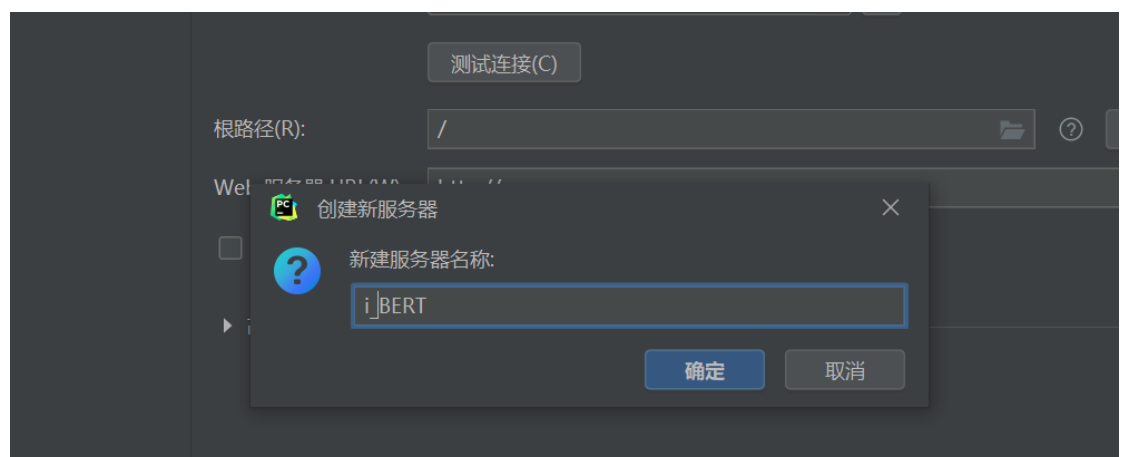
### 3、运行配置

在 pycharm 中部署远程服务器, 让远程服务器来跑我们的代码 (我的是 pycharm2022.1.3 专业版), 首先在 pycharm 中从文件夹打开 bert-master 源文件目录, 在这个文件夹下来跑模型, 这是文件之间相互引用的原因, 不在这个文件夹下跑会找不到文件。然后进行远程服务器的部署, 以下是部署的过程:

(1) 选择工具->部署->配置



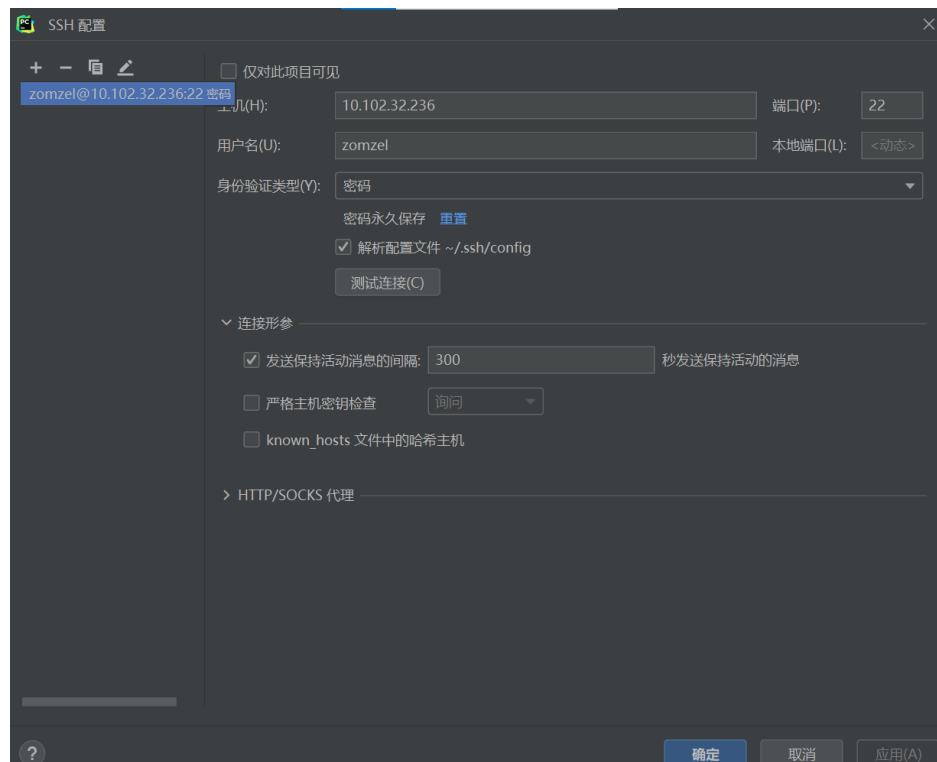
- (2) 点加号进行添加部署，选择 SFTP 类型，将服务器命名为 i\_BERT(随意)，点击确定进行下一步的配置。



- (3) 选择我们刚刚创建的服务器，点这三个点进行配置

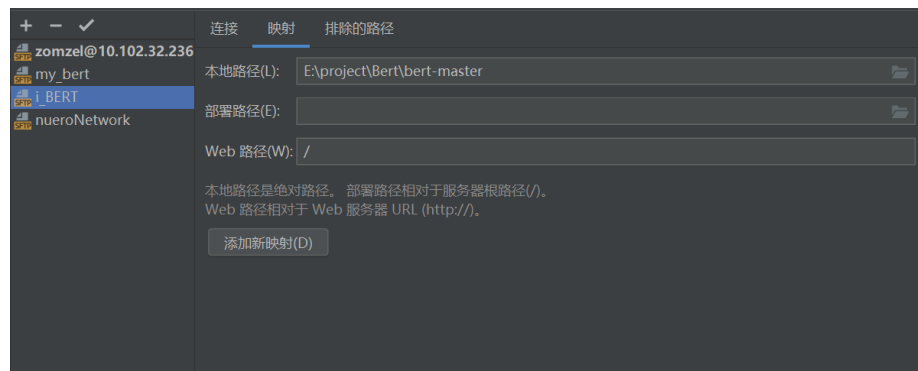


输入主机 ip、用户名，将仅对此项目可见取消勾选。点击测试连接会要求输入密码，输入密码后成功连接

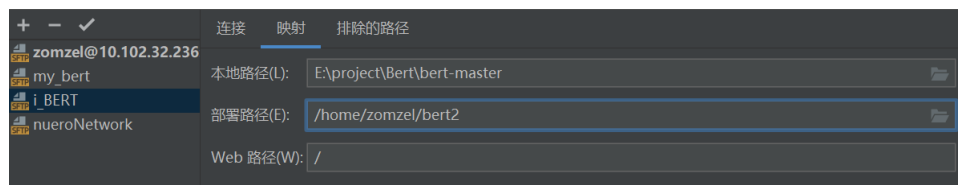




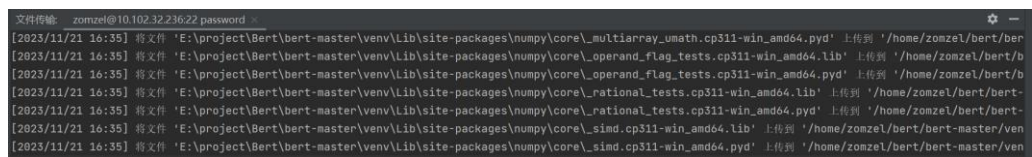
点击确定后进行路径映射的设置



本地路径是项目源码的路径也就是 bert-master 文件夹，部署路径是远程服务器上创建的 bert 文件夹的路径。Web 路径不用管，然后进行确定。



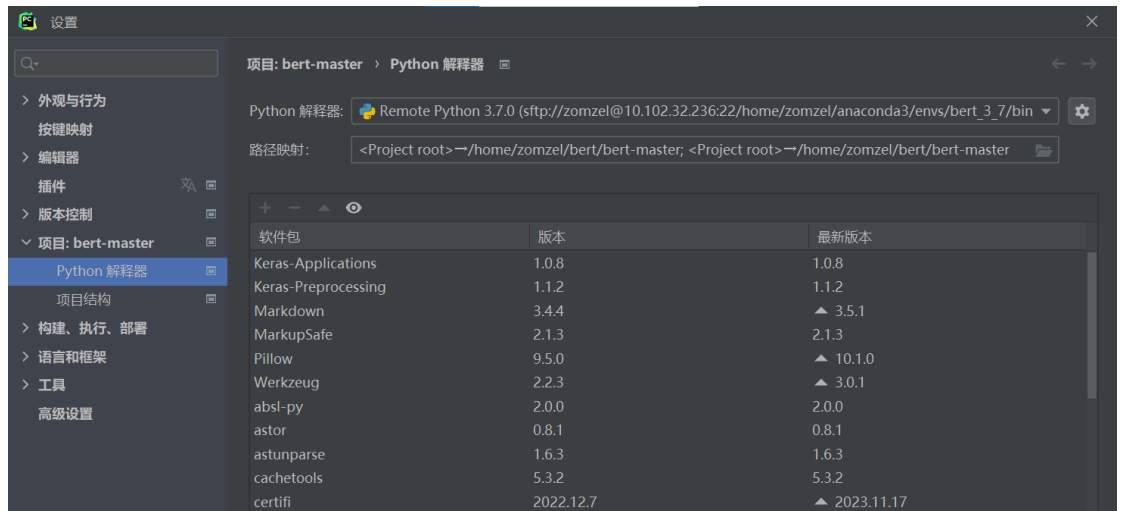
建立映射后会自动将源码文件夹中的文件上传到远程服务器中。



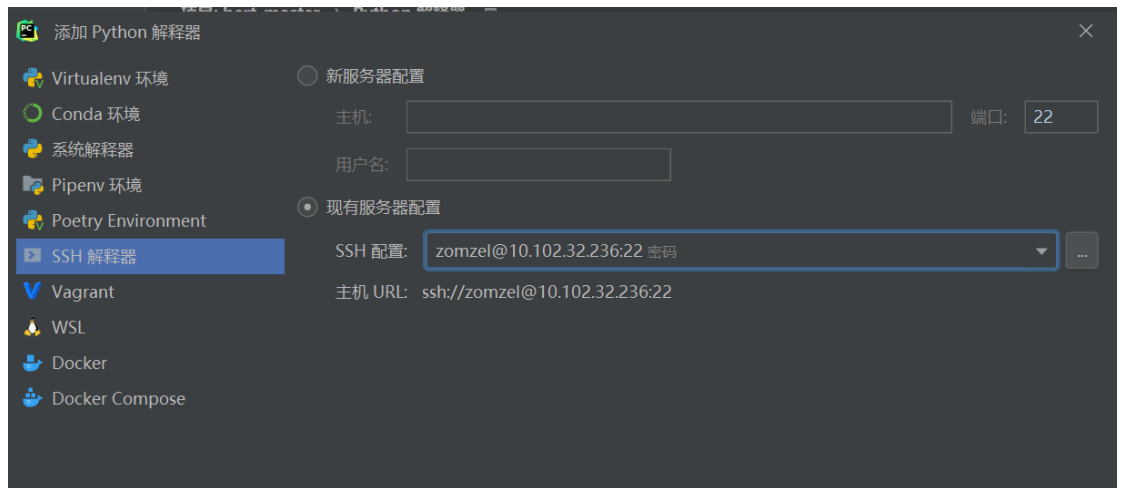
#### (4) 配置解释器

为项目代码配置解释器，文件->设置->项目->解释器

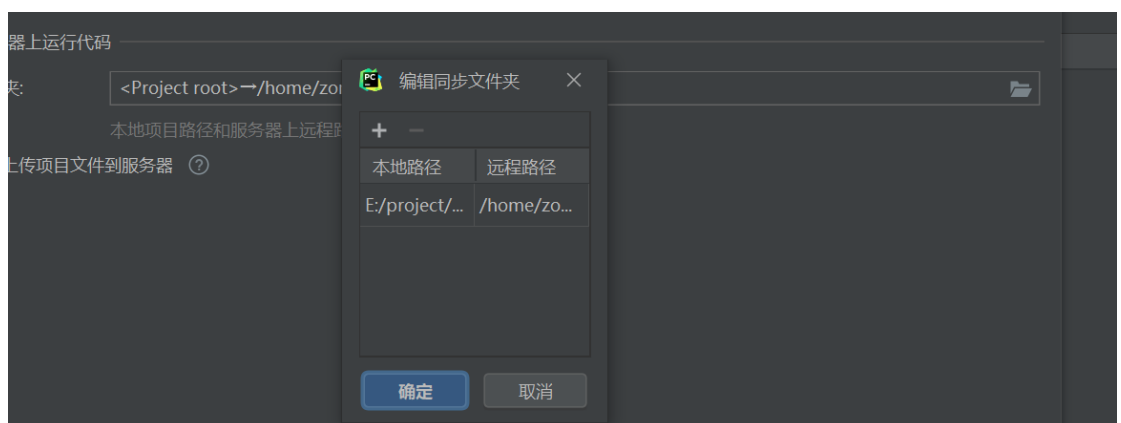


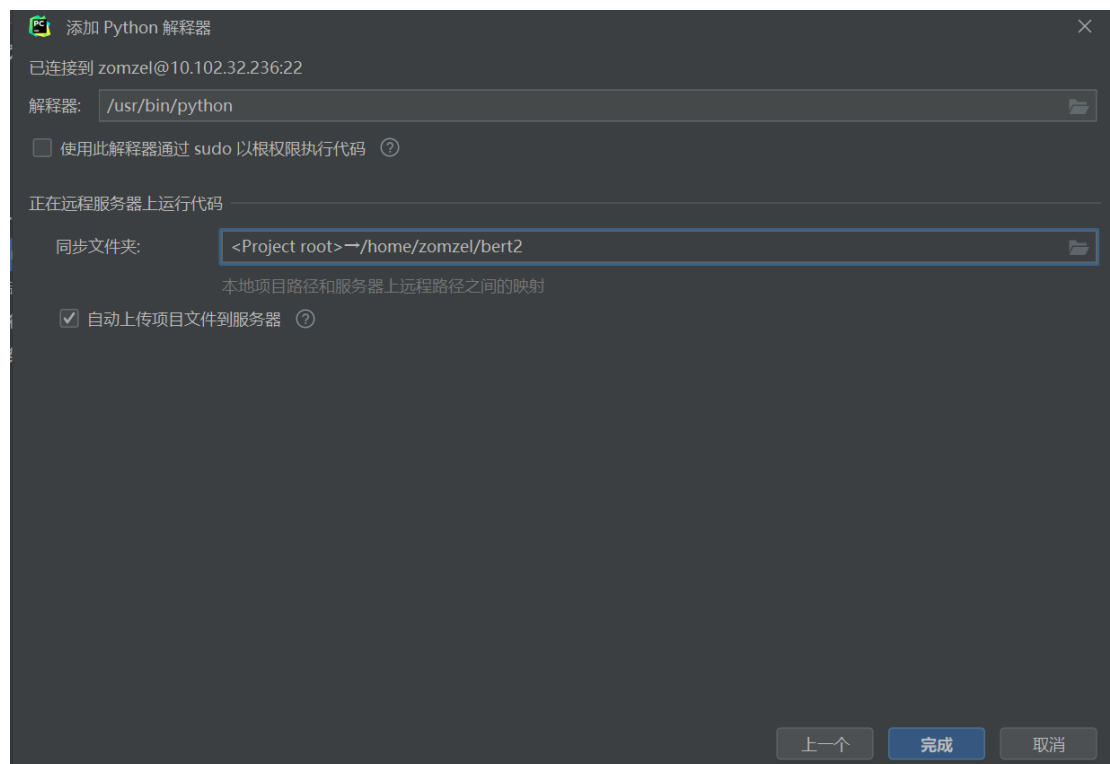


点击小齿轮添加解释器，选择 ssh 解释器和现有的远程服务器配置



点击确定下一步后会要求选择解释器的路径，也就是在远程服务器中用 conda 创建的虚拟环境的 python3.7 的文件路径。他还要求设置同步文件夹，将他默认的文件夹改成和服务部署相同的文件夹，也就是本地文件夹是 bert-master，远程文件夹是在服务器上创建的 bert 文件夹





点击完成，至此，BERT 配置完成。

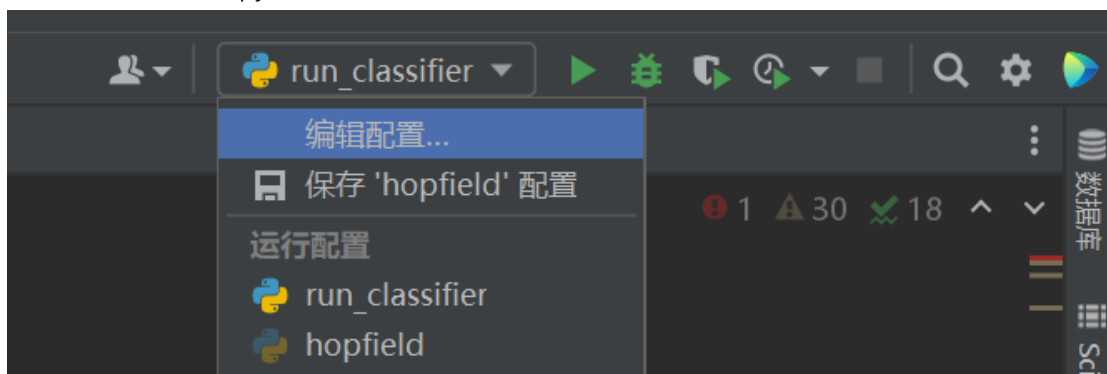
### 三、使用 BERT 预训练模型进行 Sentence 分类任务

- 1、在远程服务器上的命令行进入 bert 文件夹,此时源码文件全部上传到了该文件夹,包括 requirement.txt 文件,使用命令:

```
pip install -r requirement.txt
```

来下载所有必须的依赖包,可能有些包会下载失败,再用 pip 手动安装就行,如果还失败可以尝试下载不同的版本

- 2、配置运行参数。在 pycharm 上进行运行的配置。



运行文件选择 run\_classifier.py,然后输入运行的形参,直接复制下面的配置丢入形参配置框



第一次运行使用数据进行模型训练，输入训练参数配置

```
--task_name=MRPC
\
--do_train=true
\
--do_eval=true
\
--data_dir=../glue/glue_data/MRPC
\
--vocab_file=../glue/BERT_MODEL_DIR/uncased_L-12_H-768_A-12/vocab.txt
\
--bert_config_file=../glue/BERT_MODEL_DIR/uncased_L-12_H-768_A-12/bert_config.json
\
--init_checkpoint=../glue/BERT_MODEL_DIR/uncased_L-12_H-768_A-12/bert_model.ckpt
\
--max_seq_length=128
\
--train_batch_size=8
\
--learning_rate=2e-5
\
--num_train_epochs=1.0
\
--output_dir=../glue/outputs
```

训练得到模型后进行测试，输入下面的测试配置

```
--task_name=MRPC \
--do_predict=true \
--data_dir=../glue/glue_data/MRPC \
--vocab_file=../glue/BERT_MODEL_DIR/uncased_L-12_H-768_A-12/vocab.txt \
```

```
--bert_config_file=./glue/BERT_MODEL_DIR/uncased_L-12_H-768_A-12/bert_config.json \
--init_checkpoint= ../glue/outputs \
--max_seq_length=128 \
--output_dir=./glue/outputs/mrpc_output/
```

- 3、然后就可以直接运行 run\_classifier.py 进行训练，pycharm 会请求远程服务器的支持，第一次跑大概需要跑一个半小时，尽管输出提示信息一片红，这是正常的，如果能正常跑，请不要心急。最后的输出如下：

```
INFO:tensorflow:evaluation_loop marked as finished
I1116 21:19:41.215052 17624 error_handling.py:96] evaluation_loop marked as finished
INFO:tensorflow:***** Eval results *****
I1116 21:19:41.215052 17624 run_classifier.py:923] ***** Eval results *****
INFO:tensorflow:  eval_accuracy = 0.8382353
I1116 21:19:41.215052 17624 run_classifier.py:925]  eval_accuracy = 0.8382353
INFO:tensorflow:  eval_loss = 0.38800588
I1116 21:19:41.215052 17624 run_classifier.py:925]  eval_loss = 0.38800588
INFO:tensorflow:  global_step = 458
I1116 21:19:41.215052 17624 run_classifier.py:925]  global_step = 458
INFO:tensorflow:  loss = 0.38800588
I1116 21:19:41.215052 17624 run_classifier.py:925]  loss = 0.38800588

进程已结束,退出代码0
```

可以看到输出的正确率和损失值。

在 outputs 文件夹能找到 eval\_result.txt 文件，内容如下：

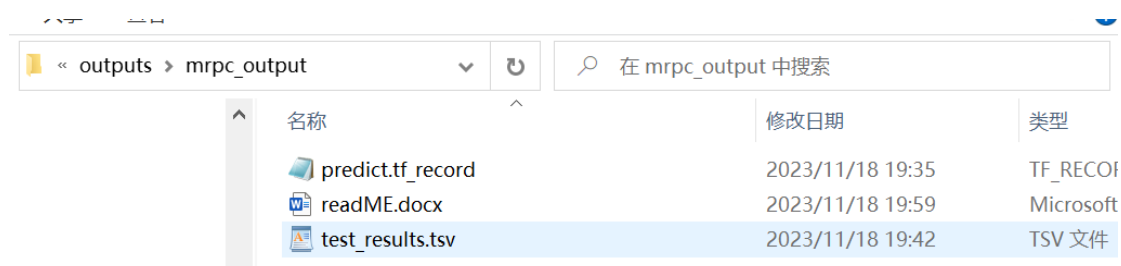
eval\_results.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

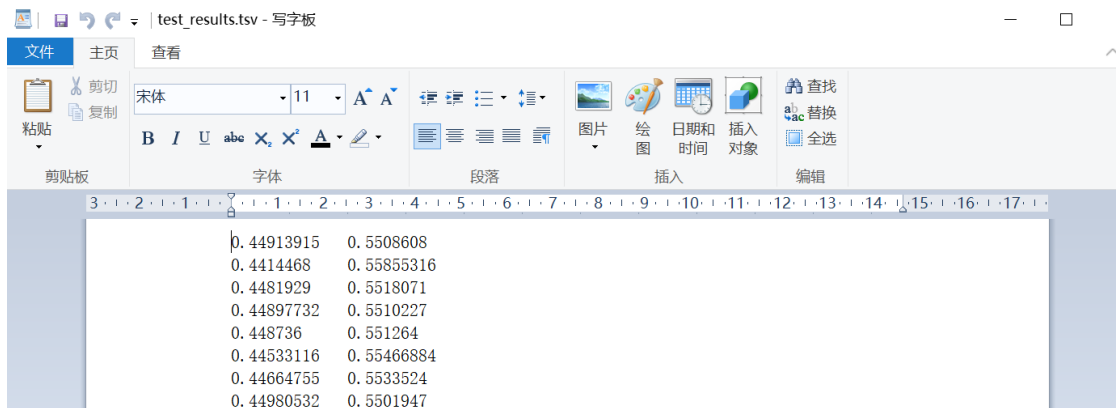
```
eval_accuracy = 0.8382353
eval_loss = 0.38800588
global_step = 458
loss = 0.38800588
```

模型训练成功！正确率和损失值与官方给出的数据相差不大。

- 4、使用训练模型和测试数据进行句子分类的预测。
- 使用测试任务的参数配置重新跑 run\_classifier.py 进行预测。（时间比训练短一些）
- 在/outputs/mrpc\_output 文件夹中找到预测输出 test\_result.tsv



其内容如下：



b.	0.44913915	0.5508608
0.	0.4414468	0.55855316
0.	0.4481929	0.5518071
0.	0.44897732	0.5510227
0.	0.448736	0.551264
0.	0.44533116	0.55466884
0.	0.44664755	0.5533524
0.	0.44980532	0.5501947

其中每一行代表一个样本属于哪个类的概率。

至此大功告成。

#### 四、总结

- 1、BERT 预训练模型可以很好的完成语言的分类和预测任务，不仅可以进行语句的分类还可以预测 mask 掉部分词的语句从而还原原本的句子，也可以预测一句话的下一句是什么。
- 2、BERT 模型具有良好的鲁棒性和可扩展性，可以很容易的通过微调将模型适应于各种语言任务，有条件的话可以试试其他的预训练模型在不同语言上的效果。