迁移学习实践

微调脚本

- 指定任务类型的微调脚本
  - huggingface已提供标准的微调脚本
- 使用微调脚本的步骤
  - 第一步: 下载微调脚本文件
    - 要现在 transformer安装包 git clone https://github.com/huggingface/transformers.git
    - cd transformers
    - pip install .
    - 去找run_glue.py
    - 属性run_glue.py文件结构
    - 注意1: examples的文件目录发生变化了

      ```
      标准化的微调脚本 run_glue.py存放位置
      /root/heimaAI/codeNLP/05fasttext及迁移学
      习/transformers/examples/pytorch/text-classification

      vim命令
      :1 到第一行
      :100 到100行
      shift+g 到最后一行

      连续使用tab在多个目录下自由跳转
      ```

    - 注意2 run_glue.py 也是有版本管理 和 transformers==2.3.0是匹配
  - 第二步: 配置微调脚本参数
    - 路径的配置 export DATA_DIR="../../../../glue_data"
    - run_glue.py 配置脚本程序
    - run_glue.sh 配置脚本

    - 第三步: 运行并检验效果
        - chmod 775 run_glue.sh 注意权限问题
        - chmod +x run_glue.sh
        - 查看文件的参数
- 如何把模型上传到hugFace平台
    - 注意: 这个是旧的创建用户,上传模型文件的方式, 现在hugFace平台已经不提供支持
        - 新的上传和下载模型的方式 把主线推完以后 再讲
    - 第一步: 在https://huggingface.co/join上创建一个帐户
        - 登录官网
    - 第二步: 在服务器终端使用transformers-cli登陆
        - transformers-cli login
    - 第三步: 使用transformers-cli上传模型并查看
        - transformers-cli ls
        - transformers-cli upload ./bert_finetuning_test/
    - 第四步: 使用pytorch.hub加载模型进行使用
        - 旧的上传方式 可以通过2.4小结的模型加载方式进行加载和使用
        - 旧的已经上传的模型, 可以继续时候

迁移学习实践1

- 迁移学习有2种方法
    - 第一种方法; 开箱即用
        - 是因为模型支持,eg bert模型 fastext模型本身在设计的时候,就考虑这种功能
        - 如果是中文的数据集, 如何与我们的模型相互结合
        - glue数据集 - nlp标准数据集
    - 第二种方法: bert + 微调网络 + 小批量标注数据 ====>完成特定的任务

- 开箱即用的迁移学习方法
- 步骤

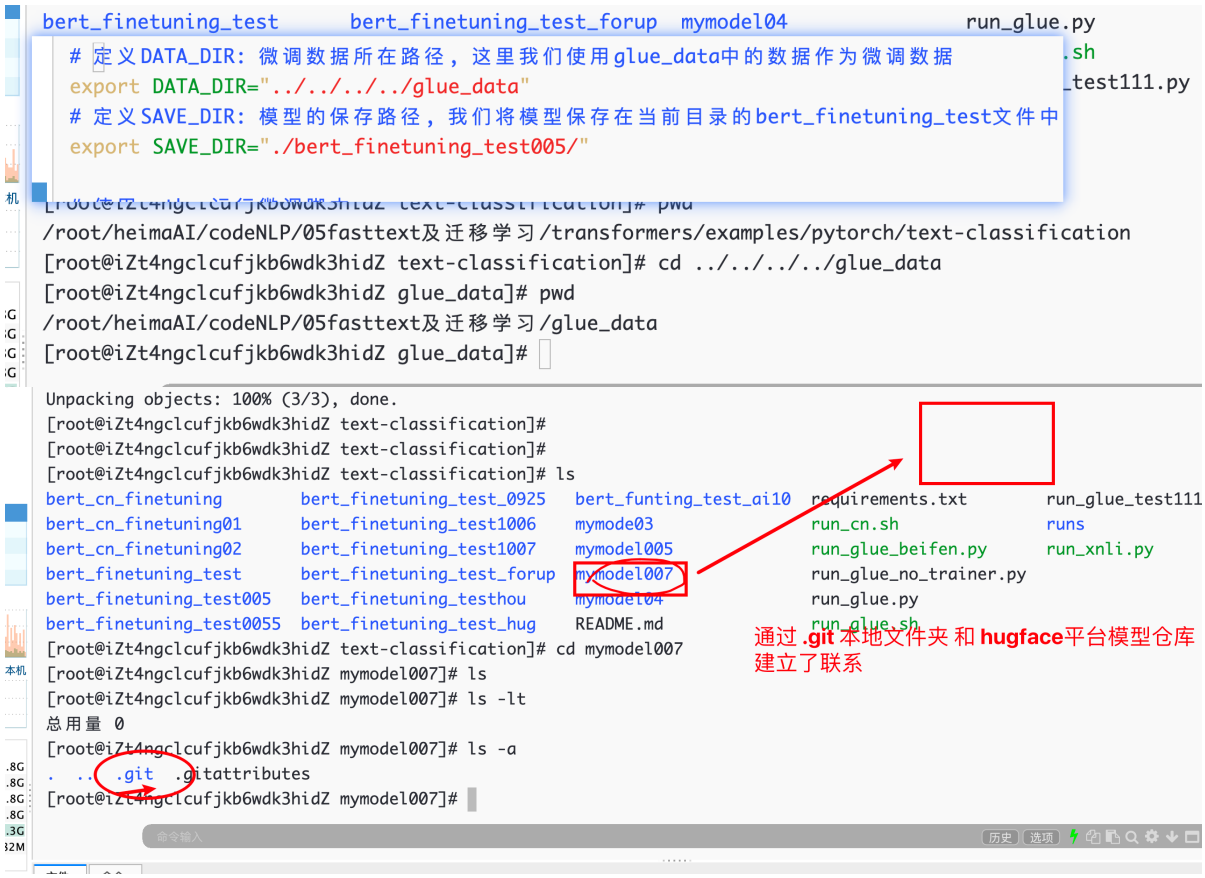
- 1 准备数据集, 格式与SST-2数据集相同!
- 2 编写 run_cn.sh文件
  - 指定数据集
  - 指定model
  - 指定任务
  - 指定训练和验证 其他参数
- 3 运行shell文件,开始训练模型
  - sh run_cn.sh ====> 一般的方法
  - Nohup ./run_cn.sh & ===> 时间比较长 训练模型的任务转成后台服务
  - 查看正在训练模型的进程号 ps -ef | grep "run_cn.sh"
- 4 上传模型
  - transformers-cli login
  - transformers-cli upload ./bert_cn_finetuning/
- 5 加载服务器第三方的模型
  - model_name = 'ItcastAI/bert_cn_finetuning'
  - 文本数值化 数值张量 提取文本分类

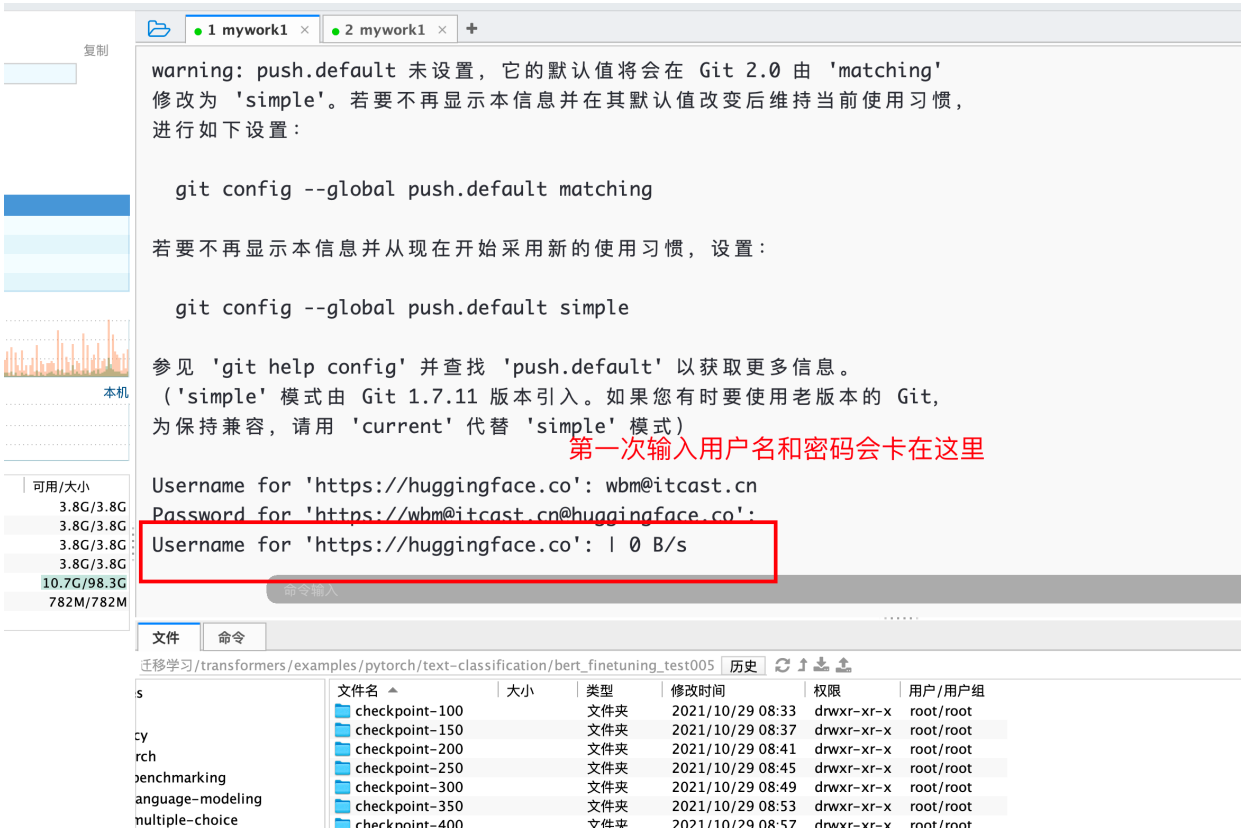
迁移学习实践2

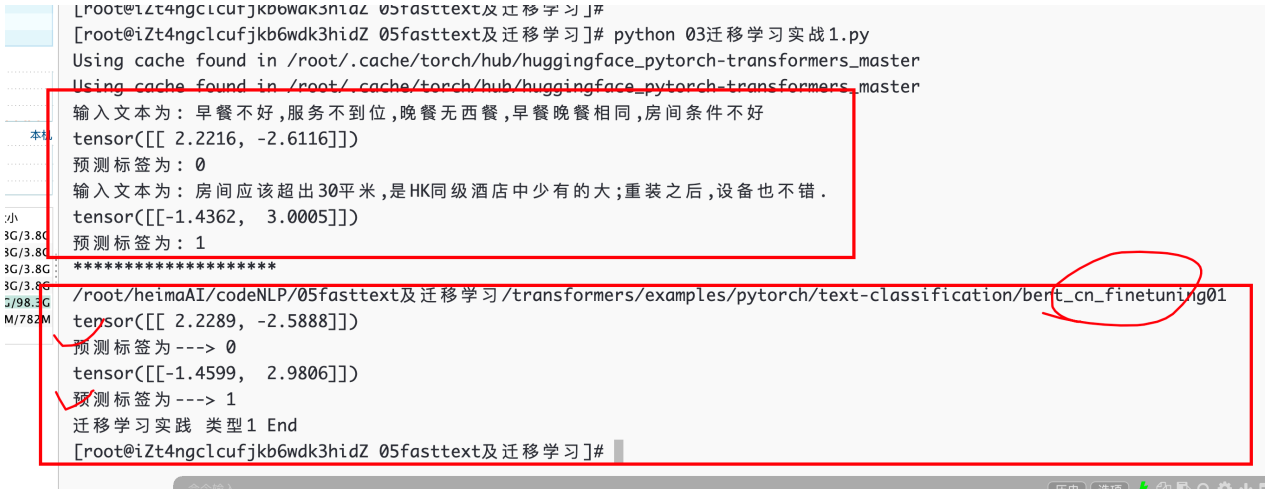
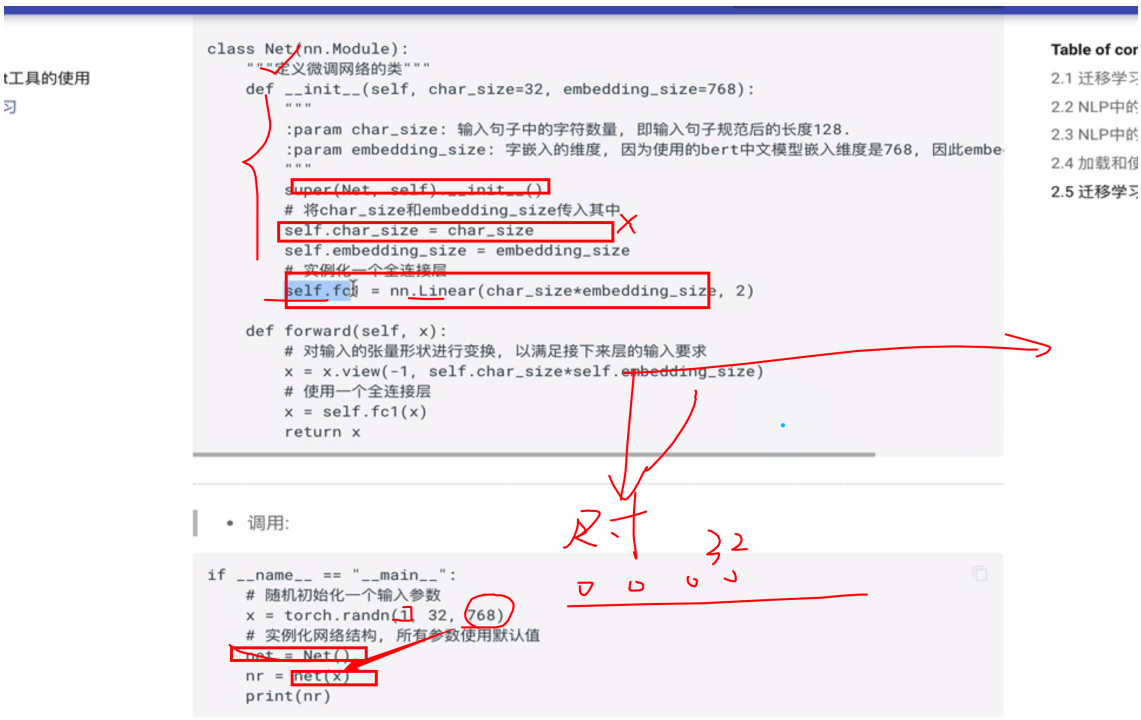
- 第二种迁移学习方法: bert + 微调网络 + 小批量标注数据 ====>完成特定的任务
  - 思路分析
    - 1 预训练模型进行特征抽取
    - 2 微调网络 (接受特征, 按照特定任务进行尺寸输出)
      - 一般2层 1000 -->600 --->10
      - 模型构建: 由**bert模型+微调网络** 组成的
    - 3 打造数据(数据预处理)
    - 4 训练模型
    - 5 模型评估
  - 步骤
    - 加载预训练模型
      - 加载不带头的预训练模型, 提取文本特征

        - 文本处理: 若长度超过32 截断, 不足32补齐
    - 新增微调网络
        - 做一个全连接层 使用了x.view(-1, **32*768**)
- 今天晚上的实验:
    - 上传模型 下载模型 本地加载
    - 开箱即用的迁移学习跑通
        - 预习: 迁移学习2

- 经典场景
    -
```
[root@iZt4ngclcufjkb6wdk3hidZ 05fasttext及迁移学习]#
[root@iZt4ngclcufjkb6wdk3hidZ 05fasttext及迁移学习]#
[root@iZt4ngclcufjkb6wdk3hidZ 05fasttext及迁移学习]#
[root@iZt4ngclcufjkb6wdk3hidZ 05fasttext及迁移学习]# find ./ -name "run_glue.py"
./transformers/examples/legacy/pytorch-lightning/run_glue.py
./transformers/examples/pytorch/text-classification/run_glue.py
./transformers/examples/tensorflow/text-classification/run_glue.py
[root@iZt4ngclcufjkb6wdk3hidZ 05fasttext及迁移学习]# pwd
/root/heimaAI/codeNLP/05fasttext及迁移学习
[root@iZt4ngclcufjkb6wdk3hidZ 05fasttext及迁移学习]#
```
    -
```
[root@iZt4ngclcufjkb6wdk3hidZ text-classification]# pwd
/root/heimaAI/codeNLP/05fasttext及迁移学习/transformers/examples/pytorch/text-classification
[root@iZt4ngclcufjkb6wdk3hidZ text-classification]# pwd
/root/heimaAI/codeNLP/05fasttext及迁移学习/transformers/examples/pytorch/text-classification
[root@iZt4ngclcufjkb6wdk3hidZ text-classification]#
[root@iZt4ngclcufjkb6wdk3hidZ text-classification]#
```
    -
```
[root@iZt4ngclcufjkb6wdk3hidZ text-classification]# pwd
/root/heimaAI/codeNLP/05fasttext及迁移学习/transformers/examples/pytorch/text-classification
[root@iZt4ngclcufjkb6wdk3hidZ text-classification]# pwd
/root/heimaAI/codeNLP/05fasttext及迁移学习/transformers/examples/pytorch/text-classification
[root@iZt4ngclcufjkb6wdk3hidZ text-classification]#
[root@iZt4ngclcufjkb6wdk3hidZ text-classification]#
[root@iZt4ngclcufjkb6wdk3hidZ text-classification]#
[root@iZt4ngclcufjkb6wdk3hidZ text-classification]# pwd
/root/heimaAI/codeNLP/05fasttext及迁移学习/transformers/examples/pytorch/text-classification
[root@iZt4ngclcufjkb6wdk3hidZ text-classification]# cd ../../../../
[root@iZt4ngclcufjkb6wdk3hidZ 05fasttext及迁移学习]# wpd
-bash: wpd: 未找到命令
[root@iZt4ngclcufjkb6wdk3hidZ 05fasttext及迁移学习]# pwd
/root/heimaAI/codeNLP/05fasttext及迁移学习
[root@iZt4ngclcufjkb6wdk3hidZ 05fasttext及迁移学习]#
```
要学会在多个目录之间 只有的跳转 使用**linux**命令的相对路径


bert_finetuning_test
bert_finetuning_test_forup
mymodel04
run_glue.py
定义DATA_DIR：微调数据所在路径，这里我们使用glue_data中的数据作为微调数据
export DATA_DIR="../../../../glue_data"
定义SAVE_DIR：模型的保存路径，我们将模型保存在当前目录的bert_finetuning_test文件中
export SAVE_DIR="./bert_finetuning_test005/"
/root/heimaAI/codeNLP/05fasttext及迁移学习/transformers/examples/pytorch/text-classification
[root@iZt4ngclcufjkb6wdk3hidZ text-classification]# cd ../../../../glue_data
[root@iZt4ngclcufjkb6wdk3hidZ glue_data]# pwd
/root/heimaAI/codeNLP/05fasttext及迁移学习/glue_data
[root@iZt4ngclcufjkb6wdk3hidZ glue_data]#
Unpacking objects: 100% (3/3), done.
[root@iZt4ngclcufjkb6wdk3hidZ text-classification]#
[root@iZt4ngclcufjkb6wdk3hidZ text-classification]#
[root@iZt4ngclcufjkb6wdk3hidZ text-classification]# ls
bert_cn_finetuning
bert_cn_finetuning01
bert_cn_finetuning02
bert_finetuning_test
bert_finetuning_test005
bert_finetuning_test0055
bert_finetuning_test_0925
bert_finetuning_test1006
bert_finetuning_test1007
bert_finetuning_test_forup
bert_finetuning_testhou
bert_finetuning_test_hug
bert_funting_test_ai10
mymode03
mymodel005
mymodel007
mymodel04
README.md
requirements.txt
run_cn.sh
run_glue_beifen.py
run_glue_no_trainer.py
run_glue.py
run_glue.sh
run_glue_test111
runs
run_xnli.py
[root@iZt4ngclcufjkb6wdk3hidZ text-classification]# cd mymodel007
[root@iZt4ngclcufjkb6wdk3hidZ mymodel007]# ls
[root@iZt4ngclcufjkb6wdk3hidZ mymodel007]# ls -lt
总用量 0
[root@iZt4ngclcufjkb6wdk3hidZ mymodel007]# ls -a
. .. .git .gitattributes
[root@iZt4ngclcufjkb6wdk3hidZ mymodel007]#
通过 .git 本地文件夹 和 hugface平台模型仓库 建立了联系


1 mywork1
2 mywork1
warning: push.default 未设置，它的默认值将会在 Git 2.0 由 'matching'
修改为 'simple'。若要不再显示本信息并在其默认值改变后维持当前使用习惯，
进行如下设置：
git config --global push.default matching
若要不再显示本信息并从现在开始采用新的使用习惯，设置：
git config --global push.default simple
参见 'git help config' 并查找 'push.default' 以获取更多信息。
('simple' 模式由 Git 1.7.11 版本引入。如果您有时要使用老版本的 Git，
为保持兼容，请用 'current' 代替 'simple' 模式)
第一次输入用户名和密码会卡在这里
Username for 'https://huggingface.co': wbm@itcast.cn
Password for 'https://wbm@itcast.cn@huggingface.co':
Username for 'https://huggingface.co': | 0 B/s
文件
命令
迁移学习/transformers/examples/pytorch/text-classification/bert_finetuning_test005
历史
文件名
大小
类型
修改时间
权限
用户/用户组
checkpoint-100 文件夹 2021/10/29 08:33 drwxr-xr-x root/root
checkpoint-150 文件夹 2021/10/29 08:37 drwxr-xr-x root/root
checkpoint-200 文件夹 2021/10/29 08:41 drwxr-xr-x root/root
checkpoint-250 文件夹 2021/10/29 08:45 drwxr-xr-x root/root
checkpoint-300 文件夹 2021/10/29 08:49 drwxr-xr-x root/root
checkpoint-350 文件夹 2021/10/29 08:53 drwxr-xr-x root/root

- 
```
[root@iZt4ngclcufjkb6wdk3hidZ 05fasttext及迁移学习]#
[root@iZt4ngclcufjkb6wdk3hidZ 05fasttext及迁移学习]# python 03迁移学习实战1.py
Using cache found in /root/.cache/torch/hub/huggingface_pytorch-transformers_master
Using cache found in /root/.cache/torch/hub/huggingface_pytorch-transformers_master
输入文本为： 早餐不好,服务不到位,晚餐无西餐,早餐晚餐相同,房间条件不好
tensor([[ 2.2216, -2.6116]])
预测标签为： 0
输入文本为： 房间应该超出30平米,是HK同级酒店中少有的大;重装之后,设备也不错.
tensor([[-1.4362,  3.0005]])
预测标签为： 1
********************
/root/heimaAI/codeNLP/05fasttext及迁移学习/transformers/examples/pytorch/text-classification/bert_cn_finetuning01
tensor([[ 2.2289, -2.5888]])
预测标签为---> 0
tensor([[-1.4599,  2.9806]])
预测标签为---> 1
迁移学习实践 类型1 End
[root@iZt4ngclcufjkb6wdk3hidZ 05fasttext及迁移学习]#
```
  - 重要场景
    - 从普通的网络层转到全连接层,为什么要view(-1,)
    - 
```
class Net(nn.Module):
    """定义微调网络的类"""
    def __init__(self, char_size=32, embedding_size=768):
        """
        :param char_size: 输入句子中的字符数量, 即输入句子规范后的长度128.
        :param embedding_size: 字嵌入的维度, 因为使用的bert中文模型嵌入维度是768, 因此embe
        """
        super(Net, self).__init__()
        # 将char_size和embedding_size传入其中
        self.char_size = char_size
        self.embedding_size = embedding_size
        # 实例化一个全连接层
        self.fc1 = nn.Linear(char_size*embedding_size, 2)

    def forward(self, x):
        # 对输入的张量形状进行变换, 以满足接下来层的输入要求
        x = x.view(-1, self.char_size*self.embedding_size)
        # 使用一个全连接层
        x = self.fc1(x)
        return x
```
    - 调用:
```
if __name__ == "__main__":
    # 随机初始化一个输入参数
    x = torch.randn(1, 32, 768)
    # 实例化网络结构, 所有参数使用默认值
    net = Net()
    nr = net(x)
    print(nr)
```

day11内容安排

1 词向量训练

2 迁移学习

概念

NLP标准数据集介绍

加载和使用预训练模型

=====>后续内容安

Day12 day13

3 使用微调脚本训练模型

训练模型 发布模型 下载模型

本地加载 (旧的方式上传的模型,本地加载是加载不了的, 旧的上传方式还是使用hub.load方式去加载)

4 迁移学习案例1

直接使用预训练模型

5 迁移学习案例2

使用 预训练模型 bert-chinese+ net+ 小批量标注数据