CNN、RNN、MLP 情感分析

环境依赖

- Python3
- PyTorch >=1.0
- TorchText
- SkLearn
- SciPy
- TensorBoardX (optional)

运行方式

- `python3 main.py --config path_to_config`
- configs文件夹中提供了一些config文件的示例，下面我也会具体说明

Config 示例

- cnn_with-static-w2v.json（带有固定的预训练词向量的CNN）：

```
{
    "sgns_model": "sgns.sogou.word",
    "dataset": "data",
    "cuda": true,
    "comment": "cnn_with-static-w2v",
    "tensorboard": true,
    "train": "sinanews.train",
    "test": "sinanews.test",
    "preload_w2v": true,
    "epoch": 50,
    "dropout": 0.5,
    "freeze": true,
    "learning_rate": 0.001,
    "train_batch_size": 128,
    "test_batch_size": 128,
```

```
    "model": "CNN",
    "loss": "cel",
    "rnn_type": "gru",
    "bidirectional": false,
    "vector_dim": 300,
    "filter_num": 256,
    "class_num": 8,
    "kernel_size": 5,
    "hidden_dim": 256,
    "rnn_layers": 2,
    "fix_length": 2500
}
```

- 参数说明：
  - sgns_model参数为预训练的Word-To-Vector模型，如果你想调用，需要在code文件夹建立名为sgns的目录，并把模型放进去，把模型的文件名作为该参数的设置，如果不使用预训练模型可以忽略该选项
  - dataset参数为数据集所在位置
  - cuda参数为是否启用CUDA加速计算，程序会根据GPU设备是否可用进行进一步判断
  - comment参数为此次运行的备注，主要用来TensorBoardX可视化
  - train为训练数据的文件名，格式同助教所给
  - test为测试数据的文件名，格式同助教所给
  - preload_w2v为是否使用预训练的模型
  - epoch为训练轮数
  - dropout，在每个模型中均加入了nn.Dropout层，该参数为dropout的比例，0为不启用
  - freeze为是否固定预训练的模型，如果不使用预训练模型可以忽略该选项
  - learning_rate为训练的学习率设置
  - train_batch_size为训练时MiniBatch的大小
  - test_batch_size为测试时MiniBatch的大小
  - model为选用的模型，可以写MLP、CNN或者RNN
  - loss为损失函数，cel为交叉熵，mse为MSE
  - rnn_type为RNN的类型，可以选择lstm或者gru，如果model选项不是RNN，可以忽略这一项

- bidirectional为RNN是否双向，如果model选项不是RNN，可以忽略这一项
- vector_dim为Embedding层向量的维数，即词向量的维数，注意如果使用预训练的词向量这里需要一致
- filter_num为CNN中filter的个数，如果model选项不是CNN，可以忽略这一项
- class_num为最后的类别数，助教的训练集为8
- kernel_size为卷积核的大小，如果model选项不是CNN，可以忽略这一项
- hidden_dim为RNN的内部的hidden_dim或为MLP中间层的节点数，如果model选项是CNN，可以忽略这一项
- rnn_layers为RNN中LSTM Cell或GRU Cell的个数，如果model选项不是RNN，可以忽略这一项
- fix_length为数据Padding到的固定长度，如果model选项不是MLP，可以忽略这一项

代码框架

代码已上传到：https://github.com/LyricZhao/EmotionAnalyzer，其中code文件夹问具体实现代码，其中大框架用PyTorch实现，用了TorchText处理数据，TensorBoardX对数据可视化，还有一些科学计算的库来辅助计算。

main.py

程序的入口，主要来加载文件，调用数据集、模型和训练的接口，在流程中传递数据。

dataset.py

加载数据集的接口，助教给的数据集格式用Tab符号分隔，也就是原生的TSV格式，我用TorchText库把文件分成Index、Label、Text三个Field，通过其前后处理的接口把数据去字母和数字，把出现次数最多的Label最为最后的Label，同时还会进行混洗等操作。

model.py

定义了CNN、RNN和MLP的模型结构，具体结构见后面的章节。

trainer.py

进行训练和测试的代码，同时还会用TensorBoardX的接口对数据进行记录和可视化。