说明

1. 生成特征文件：

1.1 生成结构编码器的特征文件 morgans.npy 和结构编码器的预训练模型

打开 MorganAE.py，首先看到：

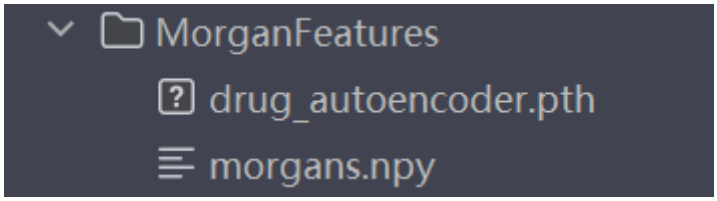
```
class Smile2Morgan():
    def __init__(self, shape=1024, radius=2):
        self.shape = shape
        self.radius = radius
        self.file = ['human/data_human.txt', 'ki/bindingDB_ki_5000.txt',
                     'dude/DUDE_train_all.txt', 'kiba/KIBA.csv', 'davis/davis_str_all.txt']
```

Self.file 就是各数据集的文件，已经给好了，不要随便动。然后，将第 198-201 行的注释取消，运行该文件，就会开始训练结构编码器：

```
194             print(f"Test Loss: {1000 * avg_test_loss}")
195         print("Training complete!")
196         torch.save(model.state_dict(), f: 'MorganFeatures/drug_autoencoder.pth')
197
198     # morgan_featurizer = Smile2Morgan()
199     # morgan_featurizer.save_morgan_feature()
200
201     # main()
```

取消注释然后运行

然后会得到两个文件：



注意：在得到这两个文件之后一定要记得把这三行代码重新注释掉，不然它会继续参与训练

1.2 生成数据集特征文件

打开 DataProcess.py，下边会有这些代码：

```
data_path = 'Drugbank'
fasta_file = 'Drugbank/data_db_protein_sequences.fasta'
PAAC_feature(fasta_file, data_path=data_path + '/data.txt', lambda_value=10, output_name=data_path+'/PAAC.tsv')
process(data_path + '/data.txt')
```

data_path 指定数据集文件夹的名字，数据集在哪个文件夹，这个变量就叫啥，名字都区分大小写

fasta_file 是指定蛋白质 fasta 文件的完整路径，格式应该是”数据集/fasta 文件名”，如果没有这个文件，代码会自动向数据集文件夹生成一个

PAAC_feature()有几个参数，第一个是 fasta 文件路径，第二个是数据集文件路径，lambda_value 是蛋白质 paac 特征的维度，output_name 是 PAAC 特征文件保存的完整路径

process()只有一个参数，就是数据集文件路径

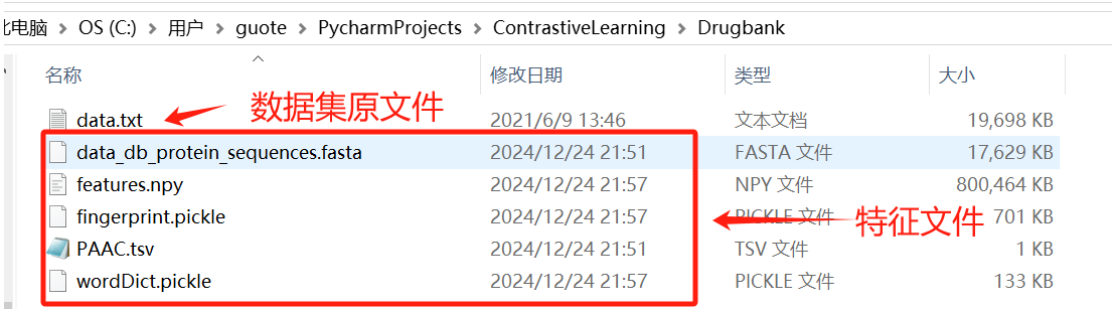
注意：

1.2.1 这个文件运行时候 numpy 版本不能太高，否则会报错。我用的版本是 1.21.5

1.2.2 比如 human 数据集的数据文件是 human/data_human.txt，那么上边的 data_path=human，数据集文件路径=data_path+’data_human.txt’。

1.2.3 另外 lambda_value 不能随便填，PAAC 特征的维度是 20+lambda，如果 lambda_value=10，PAAC 特征维度就是 30，这个 lambda_value 必须要比数据集中最小的氨基酸序列长度小 1，所以 Drugbank 数据集的 lambda_value 最大只能是 10（最短的序列长度是 11），另外还有一个数据集最大也只能是 10，我忘了是哪个，其他数据集我设置的都是 30。如果 lambda_value+1 的值比数据集中最短的序列长度值还大，生成的 paac.tsv 文件就是空的，记得检查。

如果没有问题，数据集文件夹下就会出现多出这几个文件：



2.运行

在 main.py 中，有很多参数可选：

```
parser = argparse.ArgumentParser()
parser.add_argument( *name_or_flags: "--lr", help="learning rate", type=float, default=0.005)
parser.add_argument( *name_or_flags: "--ld", help="learning rate decay", type=float, default=0.5)
parser.add_argument( *name_or_flags: "--epoch", help="epoch", type=int, default=30)
parser.add_argument( *name_or_flags: "--features", help="feature dimension", type=int, default=40)
parser.add_argument( *name_or_flags: "--GNN_depth", help="gnn layer number", type=int, default=3)
parser.add_argument( *name_or_flags: "--MLP_depth", type=int, default=2)
parser.add_argument( *name_or_flags: '--weight_decay', type=float, default=1e-6, help='Weight decay (L2 loss on parameters).')
parser.add_argument( *name_or_flags: "--dataset", type=str, default='human')
parser.add_argument( *name_or_flags: "--batch_size", type=int, default=256)
parser.add_argument( *name_or_flags: "--lambd1", help="weight for prediction loss", type=float, default=1.0)
parser.add_argument( *name_or_flags: "--lambd2", help="weight for 2nd stage contrastive loss", type=float, default=1.0)
parser.add_argument( *name_or_flags: "--lambd3", help="weight for 1st stage drug's intra-modality contrastive loss", type=float,
                    default=0.1)
parser.add_argument( *name_or_flags: "--lambd4", help="weight for 1st stage protein's intra-modality contrastive loss", type=flo
                    default=0.1)
parser.add_argument( *name_or_flags: "--resume", type=bool, help="resume training from the checkpoint", default=False)
```

换数据集就是--dataset 参数，lambd1,lambd2,lambd3,lambd4 是几个损失的权重，后边两项应该小一点，因为很难收敛，第二项看情况。

e.g. : python main.py --dataset human --epoch 50 --lambd2 0.3

我设置了每 20 轮就给模型保存一次，保存到 checkpoints 文件夹下，重复运行会覆盖模型文件，注意备份。如果想接着之前的继续训练，就把--resume 设置为 True，代码会自动从文件夹中加载最后保存的模型