package

python file

/basedline/model: (model for training the accelerometer data)

- __init__.py
  - initialization of model
  - time window size should be correct in forward() of class SegmentationFusionModel
- accel.py
  - It contains model components.

/baseline

- testTrain.py
  - main function for both training and testing
  - model is saved in training stage.
    - torch.save(model.state_dict(), "modelname.pt")
  - testing is used specified saved model.
    - model.load_state_dict(torch.load("modelname.pt"))
- train.py
  - training and testing functions.
  - number of training epochs specify here.
  - training batch size is specified here.
  - time window size should be correct
  - If you want to modify the subfunction to evaluate other results in each train step, each train epoch end, as well as in testing. please look at https://lightning.ai/docs/pytorch/stable/model/build_model_advanced.html

/data/

- all_test_pkl
  - different time window pkl data for experiment 1
- successful_test_pkl
  - different time window pkl data for experiment 2
- train_pkl
  - different time window pkl training data
- unsuccessful_test_pkl/all_unsuccessful
  - different time window pkl data for experiment for experiment 3
- unsuccessful_test_pkl/start
  - different time window pkl data for experiment for experiment 4
- unsuccessful_test_pkl/continue
  - different time window pkl data for experiment for experiment 5

/data_loading/

- dataset.py
  - this class is used to create training and testing dataset, which is initialized by pkl files and corresponding extractor in testTrain.py:

```
# extract data based on features selected
ds = FatherDataset(examples, extractors)
test_ds = FatherDataset(test_examples, extractors)
```

- extractors.py
  - This class is used to initialize an extractor based on the selected modality. In our case, it is used to extract data from the accelerometer corresponding to a specific time period, based on the start and end times of the created sample.

- make_examples.py
  - It is used to create a pkl file by using the created CSV file of samples.

- utiles.py
  - the functions are used in make_examples.py
  - the content of pkl files is defined here, for example

```
examples.append({
    'id': example_id,
    'pid': i,
    'ini_time': all_ini_time,
    'end_time': all_end_time,
    # data
    'vad': unsuccessful_temp_vad
})
```

/preprocess/audio/

- all_label
  - different time window label of experiment 1
- all_sample
  - samples in csv format for experiment 1
- filter_vad
  - According to our filtering rules, generate based on the original VAD file.
- successful_train_samples
  - training samples in csv format
- successful_train_ground_truth
  - label of training samples

- successful_test_samples
    - test samples in csv format of experiment 2
- successful_test_ground_truth
    - label of experiment 2
- unsuccessful_intention_test_sample/
    - test samples in csv format of experiment 3, 4, 5
- unsuccessful_intention_test_label
    - label of experiment 3,4,5
- generate_samples.py
    - it is to generate samples and labels in csv format by using filtered VAD files.