ASR实验roadmap

- 配置config文件目录（推荐使用yaml格式）
  1. data.yaml 音频和文本的包括**audio**：特征类型fbank/MFCC、特征维度、是否计算cmvn；**text**：embedding维度、是否提取子词subword和所需vocab位置；batch_size、epoch和数据所在路径等
  2. model.yaml 这一部分定义模型结构和参数，具体因模型种类不同而异，可包括如：LAS中encoder、decoder的层数，是否为双向LSTM编解码，dropout参数，norm方法，模型的维度，attention的维度和head数，CTC权重等；语言模型训练参数

- 数据预处理preprocessing
  1. audio.py Input： config/args.yaml 从路径中加载音频数据，定义**ExtractFeature**类 return: features [sequence_length, f_dim]
  2. text.py Input: config/args.yaml 从路径中加载文本数据，定义tokenizer做子词分割部分做word embedding和padding等操作 return: embedding vector: [sequence_length, emb_dim]

- 模型搭建部分module.py

1. encoder: prenet vgg部分，编码器neural network架构
2. attention： 类型和维度 如locationAwareattention
3. decoder：自回归特性单向LSTM，输入为文本embedding和attention结果，输出端为cross-entropy loss
4. LM：语言模型常规方法训练
5. （可选）CTC：构建CTC state按照文章定义PrefixScore用来与att decoding结果联合训练

- 执行脚本
  1. 用来执行各类任务的**solver**：加载config，backward，加载预训练模型，打印提示信息等，后续脚本均继承该类即可
  2. 训练语言模型**train_lm.py**：从配置文件的路径中加载数据或预训练模型以及optimizer，进行训练和验证，保存模型并同时log在tensorboard中
  3. 训练ASR模型**train_asr.py:**
     - Training过程
       1. DataLoader加载数据，构建train_loader和test_loader
       2. 定义train函数在one-epoch上的过程，model(x)→计算loss→loss.backward()→optimizer.step()→print(loss)
     - 验证步骤validate同时比较dev_loss更新loss存储best-ckpt，打印log信息
  4. Test过程**test_asr.py**
     1. BeamDecoder定义
     2. 设定模型，加载测试数据
     3. CER/WER计算
- 最终命令行训练和测试脚本main.py

  从命令行读取训练语言模型还是ASR，亦或是训练/测试 ASR模型