LAS

参考实现

run

data

model

Seq2Seq

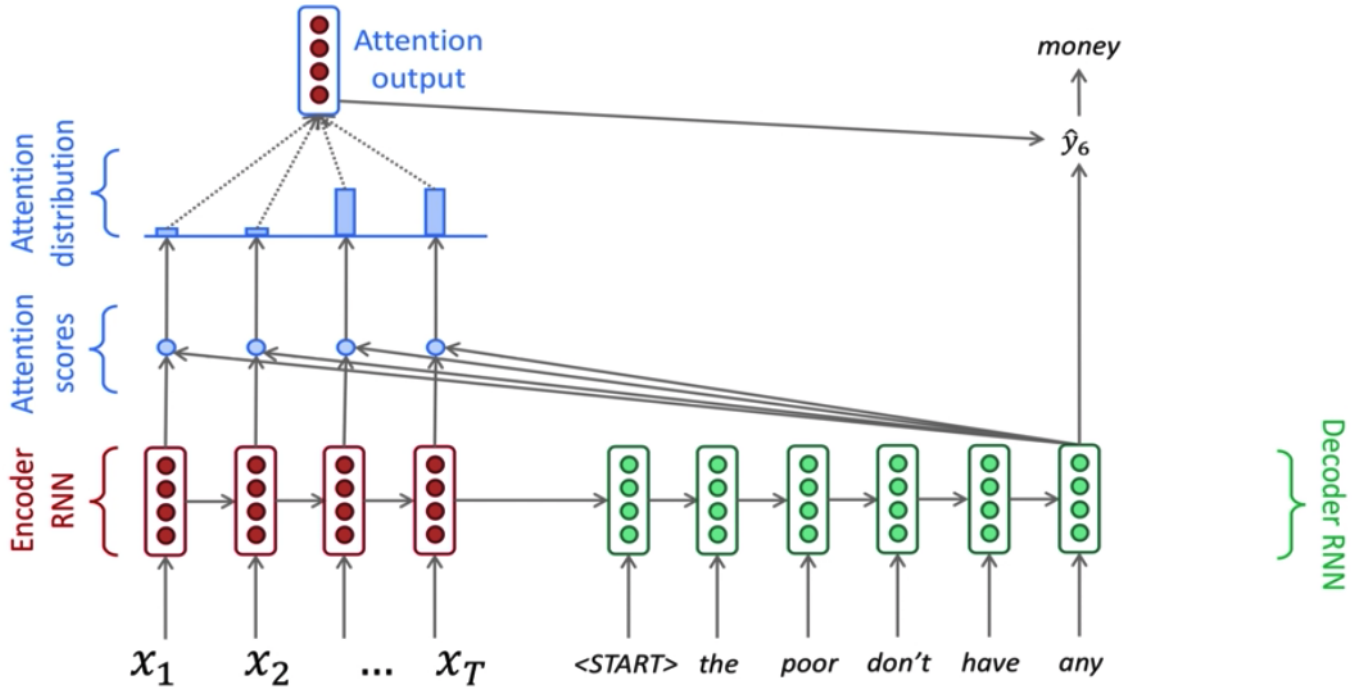
Solver

train

recognize

attention

将seq2seq的输入换成语音特征序列 $X = x_1, x_2, ..., x_T$ ，输出换成文本序列 $Y = y_1, y_2, ..., y_U$ ，可直接将X映射成Y，从而实现end2end asr。



参考实现

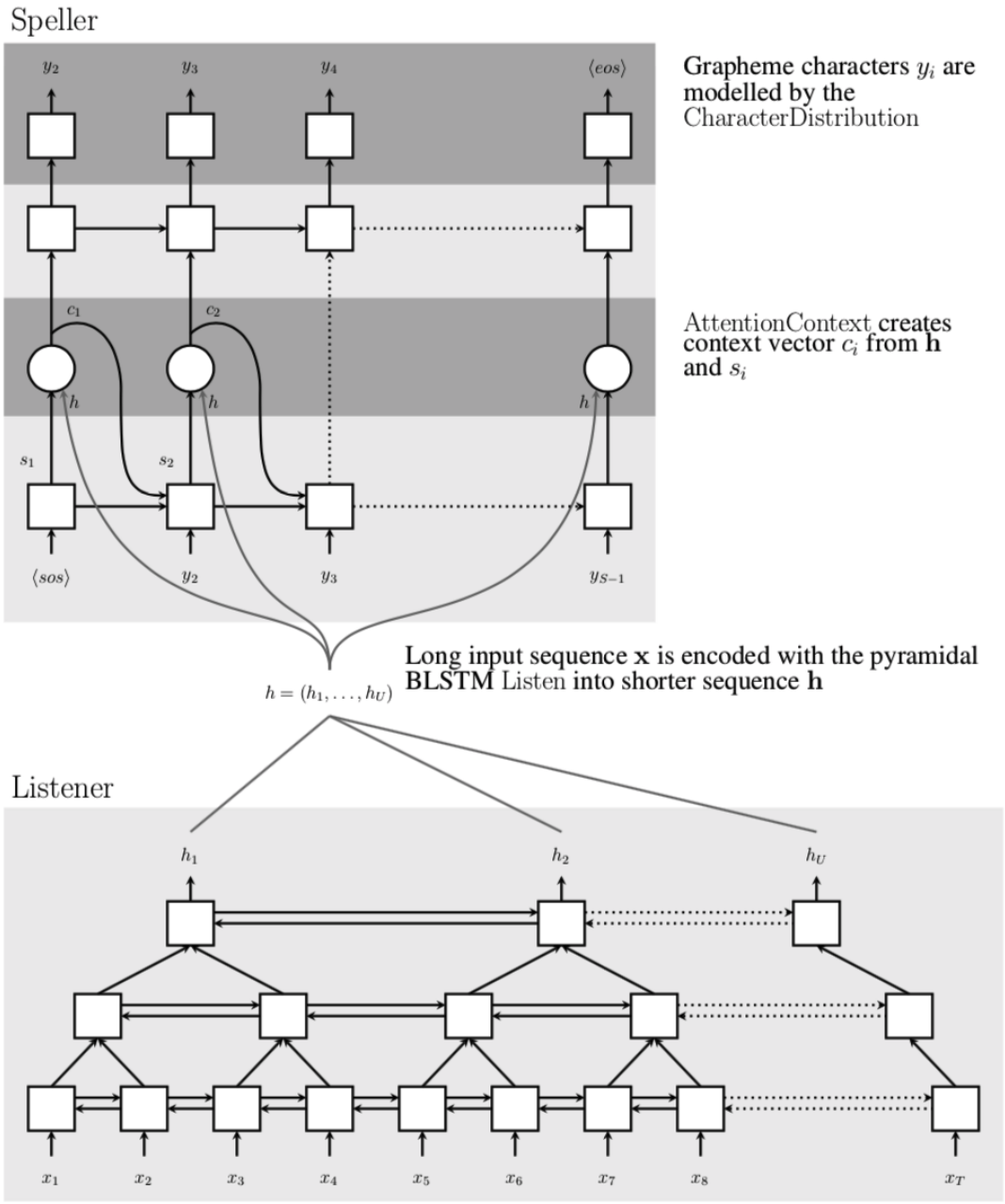


Figure 1: Listen, Attend and Spell (LAS) model: the listener is a pyramidal BLSTM encoding our input sequence $\mathbf{x}$ into high level features $\mathbf{h}$, the speller is an attention-based decoder generating the $\mathbf{y}$ characters from $\mathbf{h}$.

https://github.com/kaituoxu/Listen-Attend-Spell.git

run

- step0 数据准备，生成文件如下

  test文件

```Shell
BAC009S0002W0122 而对楼市成交抑制作用最大的限购
BAC009S0002W0123 也成为地方政府的眼中钉
BAC009S0002W0124 自六月底呼和浩特市率先宣布取消限购后
BAC009S0002W0125 各地政府便纷纷跟进
BAC009S0002W0126 仅一个多月的时间里
BAC009S0002W0127 除了北京上海广州深圳四个一线城市和三亚之外
BAC009S0002W0128 四十六个限购城市当中
BAC009S0002W0129 四十一个已正式取消或变相放松了限购
BAC009S0002W0130 财政金融政策紧随其后而来
BAC009S0002W0131 显示出了极强的威力
BAC009S0002W0132 放松了与自往需求密切相关的房贷政策
BAC009S0002W0133 其中包括对拥有一套住房并已结清相应购房贷款的家庭
BAC009S0002W0134 为改善居住条件再次申请贷款购买普通商品住房
```

wav.scp文件

```Shell
BAC009S0002W0122 /data2/ASR_data_opensource/aishell/data_aishell/wav/train/S0002/BAC009S0002W0122.wav
BAC009S0002W0123 /data2/ASR_data_opensource/aishell/data_aishell/wav/train/S0002/BAC009S0002W0123.wav
BAC009S0002W0124 /data2/ASR_data_opensource/aishell/data_aishell/wav/train/S0002/BAC009S0002W0124.wav
BAC009S0002W0125 /data2/ASR_data_opensource/aishell/data_aishell/wav/train/S0002/BAC009S0002W0125.wav
BAC009S0002W0126 /data2/ASR_data_opensource/aishell/data_aishell/wav/train/S0002/BAC009S0002W0126.wav
```

utt2spk文件

```
BAC009S0002W0493 S0002
BAC009S0002W0494 S0002
BAC009S0002W0495 S0002
BAC009S0003W0121 S0003
BAC009S0003W0122 S0003
BAC009S0003W0123 S0003
```

spk2utt文件

```
S0002 BAC009S0002W0122 BAC009S0002W0123 BAC009S0002W0124 BAC009S0002W0125
BAC009S0002W0126 BAC009S0002W0127 BAC009S0002W0128 BAC009S0002W0129
BAC009S0002W0130 BAC009S0002W0131 BAC009S0002W0132 BAC009S0002W0133
BAC009S0002W0134 BAC009S0002W0135 BAC009S0002W0136 BAC009S0002W0137
BAC009S0002W0138
S0003 BAC009S0003W0121 BAC009S0003W0122 BAC009S0003W0123 BAC009S0003W0124
BAC009S0003W0125 BAC009S0003W0126 BAC009S0003W0127 BAC009S0003W0128
BAC009S0003W0129 BAC009S0003W0130 BAC009S0003W0131 BAC009S0003W0132
BAC009S0003W0133 BAC009S0003W0134 BAC009S0003W0135 BAC009S0003W0136
BAC009S0003W0137 BAC009S0003W0138 BAC009S0003W0139 BAC009S0003W0140
BAC009S0003W0141 BAC009S0003W0142 BAC009S0003W0143 BAC009S0003W0144
BAC009S0003W0145 BAC009S0003W0146
```

- step1 特征生成
  - 计算fbank

    compute-fbank-feats 输入为wav.scp文件，输出到标准输出。首先读取wav文件内容到Vector中，然后调用Fbank计算特征，输出为Matrix。

    copy-feats 输入为上一步的标准输出，将Matrix输出到archive or script file。
  - 计算倒谱均值和方差

    compute-cmvn-stats 输入为feat.scp文件，输出为cmvn.scp
  - dump特征

    apply-cmvn调用ApplyCmvn对feature进行计算
- step2 词典和json文件准备
  - 词典文件

```
<unk> 0
<sos> 1
<eos> 2
一 3
丁 4
七 5
万 6
丈 7
三 8
上 9
下 10
不 11
与 12
丐 13
```

- json文件

Shell | 复制代码


```
{
    "utts": {
        "BAC009S0002W0122": {
            "input": [
                {
                    "feat": "/data/wll123605/workspace/asr/Listen-Attend-Spell/egs/aishell/dump/train/deltatrue/feats.1.ark:17",
                    "name": "input1",
                    "shape": [
                        598,
                        240
                    ]
                }
            ],
            "output": [
                {
                    "name": "target1",
                    "shape": [
                        15,
                        4233
                    ],
                    "text": "而对楼市成交抑制作用最大的限购",
                    "token": "而 对 楼 市 成 交 抑 制 作 用 最 大 的 限 购",
                    "tokenid": "2996 1012 1892 1122 1380 83 1427 357 1682479 1741 815 2554 3968 3555"
                }
            ],
            "utt2spk": "S0002"
        },
        "BAC009S0002W0123": {
            "input": [
                {
                    "feat": "/data/wll123605/workspace/asr/Listen-Attend-Spell/egs/aishell/dump/train/deltatrue/feats.1.ark:145495",
                    "name": "input1",
                    "shape": [
                        385,
                        240
                    ]
                }
            ],
            "output": [
                {
                    "name": "target1",
                    "shape": [
```

```
                        11,
                        4233
                    ],
                    "text": "也成为地方政府的眼中钉",
                    "token": "也 成 为 地 方 政 府 的 眼 中 钉",
                    "tokenid": "59 1380 37 723 1658 1622 1170 2554 2597
31 3838"
                }
            ],
            "utt2spk": "S0002"
        }, ...
    }
}
```

- step3 网络训练

Shell | 复制代码

```
train.py --train_json dump/train/deltatrue/data.json --valid_json
dump/dev/deltatrue/data.json --dict data/lang_1char/train_chars.txt --
einput 240 --ehidden 256 --elayer 3 --edropout 0.2 --ebidirectional 1 --
etype lstm --atype dot --dembed 512 --dhidden 512 --dlayer 1 --epochs 20
--half_lr 1 --early_stop 0 --max_norm 5 --batch_size 32 --maxlen_in 800 -
-maxlen_out 150 --optimizer adam --lr 1e-3 --momentum 0 --l2 1e-5 --
save_folder
exp/train_in240_hidden256_e3_lstm_drop0.2_dot_emb512_hidden512_d1_epoch20
_norm5_bs32_mli800_mlo150_adam_lr1e-3_mmt0_l21e-5_delta --checkpoint 0 --
continue_from "" --print_freq 10 --visdom 0 --visdom_id "LAS Training"
```

- step4 解码和打分

Shell | 复制代码

```
recognize.py --recog_json dump/test/deltatrue/data.json --dict
data/lang_1char/train_chars.txt --result_label
exp/train_in240_hidden256_e3_lstm_drop0.2_dot_emb512_hidden512_d1_epoch20
_norm5_bs32_mli800_mlo150_adam_lr1e-3_mmt0_l21e-
5_delta/decode_test_beam30_nbest1_ml100/data.json --model_path
exp/train_in240_hidden256_e3_lstm_drop0.2_dot_emb512_hidden512_d1_epoch20
_norm5_bs32_mli800_mlo150_adam_lr1e-3_mmt0_l21e-5_delta/final.pth.tar --
beam_size 30 --nbest 1 --decode_max_len 100
```

data

- AudioDataset
  - 输入为data.json文件中的utts，按照每个数据的shape[0]进行排序
  - 根据最长的输入输出数据计算分组factor
  - 根据factor进行分组，生成mini batch
- AudioDataLoader
  - load_inputs_and_targets 使用kald_io读取特征，按照特征长度进行排序，移除输出长度为0数据
  - collate_fn 对数据进行填充
  - 最终输出为xs_pad、ilens、ys_pad，分别表示padded后的特征数据、特征数据的长度、目标label，shape分别为N * Ti * D、N、N * To

model

Shell 复制代码

```
Seq2Seq(
  (encoder): Encoder(
    (rnn): LSTM(240, 256, num_layers=3, batch_first=True, dropout=0.2, bidirectional=1)
  )
  (decoder): Decoder(
    (embedding): Embedding(4233, 512)
    (rnn): ModuleList(
      (0): LSTMCell(1024, 512)
    )
    (attention): DotProductAttention()
    (mlp): Sequential(
      (0): Linear(in_features=1024, out_features=512, bias=True)
      (1): Tanh()
      (2): Linear(in_features=512, out_features=4233, bias=True)
    )
  )
)
```

- encoder
  - 3层blstm 输入向量为240维，隐藏层为256维，dropout为0.2
  - forward 输入为padded_input，shape为N * T * D，N为batch size，T为frame数，D为特征长度。另一个输入input_lengths为长度N的list tensor，每个元素为特征长度。对padded_input

进行pack_padded_sequence，作为blstm的输入。输出再进行pad_packed_sequence处理，输出shape为N * T * H

- decoder
  - 1层双向LSTMCell，输入维度为vocab embedding维度加上encoder编码的特征输出，一般与隐藏层节点数目相同，隐藏层节点数目为512
  - 一个attention
  - 一个mlp，由线性层、tanh、线性层组成。第一个线性层的输入维度为encoder_hidden_size + hidden_size，输出为hidden_size，第二个线性层的输入为hidden_size，输出为vocab size
  - forward 输入padded_input为标签y，shape为N * To，N为batch size，To为输出标签y的长度。另一个输入encoder_padded_outputs为encoder的输出，shape为N * T * H。
    - 首先构建ys_in和ys_out，分别lstm的输入和计算交叉墒，shape为N * T，输出标签长度为T
    - 初始化lstm的h和c、attention向量
    - 开始循环，遍历输出标签长度T
      - 对ys_in进行padded，然后通过embedding层，和初始attention向量拼接在一起作为lstm的输入
      - 进行rnn计算，获得h和c，最后一层的输出h作为rnn的输出
      - 进行attention计算，获得attention向量和权重
      - 将attention向量和lstm输出h拼接在一起，作为mlp的输入
      - 进行mlp计算获得最终的输出标签，存放到y_all列表中
    - 对y_all和ys_out_pad计算交叉墒，获得loss
  - recognize_beam
    - 输入为encoder_outputs，char_list和args，分别为声学特征的编码(shape为T * D)、所有字符列表以及beam（用于获得nbest个结果）
    - 初始化decoder的h和c、attention向量
    - 准备初始时刻label y为sos、hyps列表中存放score为0的label sos，其中yseq列表存放已经预测出来的label，c_prev存放前一步的c，h_prev存放前一步的h，a_prev存放前一步的attention向量
    - 循环encoder_outputs的长度，即遍历所有frame
      - 循环遍历hyps，取出当前步的decoder输入hyp['yseq'][i]做一个embedding，然后和前一步的attention向量拼接在一起，送入rnn进行计算，对每一层rnn进行同样处理
        - 得到的最后一层的输出h作为rnn的输出
        - 进行attention计算，获得attention向量
        - 将rnn的输出和attention向量拼接在一起作为mlp的输入，进行mlp计算
        - 将mlp的结果进行softmax计算，获取得分最高的nbest结果
        - 更新hpys中yseq、h_prev、c_prev、a_prev和score

            - 如果到了最后一帧，在yseq中放入eos
            - 将nbest结果分为两类，一类是遇到eos预测结束，一类是仍然需要继续预测的继续进行循环处理
        - 最后将nbest结果按照score得分排序输出
- pack_padded_sequence和pad_packed_sequence
  https://blog.csdn.net/yaohaishen/article/details/120222046
- Embedding
  https://www.jianshu.com/p/63e7acc5e890

Seq2Seq

将encoder和decoder组装成一个model，提供load_model、forward、recognize、serialize接口。

- load_model 调用torch.load加载模型
- forward分别调用encoder和decoder的forward方法，输入问padded_input、input_lengths、padded_target，输出为loss值
- recognize调用encoder的forward方法进行编码，然后调用decoder的recognize_beam进行解码，输入input的shape为T * D，input_length为特征长度，char_list为字符列表，args.beam
- serialize将模型的参数序列化成json格式

Solver

包含data loader、model、optimizer、training config、save and load model参数。

- train 循环进行epochs步训练，首先打开model的train模式，然后调用run_one_epoch进行一步训练，必要时保存模型，关闭model的train模式，并调用run_one_epoch进行验证，获得loss，如果loss大于前一步的loss，则调整学习率。最后保存loss最低的模型。
- run_one_epoch 如果参数cross_valid为true则使用验证集进行验证，否则使用训练集进行训练
    - 循环遍历一批数据
    - 获得输入特征数据、特征长度列表和目标label数据，并移到gpu上
    - 调用model计算loss值
    - 如果是训练，调用optimizer计算
    - 最后返回这一批数据的平均loss值

train

- 使用AudioDataset和AudioDataLoader加载数据
- 获取词典、sos_id和eos_id
- 分别创建encoder和decoder

- 将encoder和decoder加入Seq2Seq中，创建model
- 创建优化器
- 创建solver
- 调用solver的train方法进行训练

recognize

- 调用Seq2Seq的load_model方法加载模型
- 获取字典、sos_id和eos_id
- 加载要识别的json格式文本
- 循环遍历所有音频进行识别
- 首先通过kaldi_io读取音频特征
- 接着调用Seq2Seq的recognize方法进行解码

attention

Given a set of vector values, and a vector query, attention is a technique
 to compute a weighted sum of the values, dependent on the query.
这里query为decoder的输出，shape为N * To * H，values为encoder的编码输出，shape为N * Ti * H

- forward
  - query和values的转置进行点积 (N, To, H) * (N, H, Ti) –> (N, To, Ti)
  - 进行softmax得到概率分布(N, To, Ti)
  - 概率分布和values做点积得到attention输出向量(N, To, Ti) * (N, Ti, H) –> (N, To, H)