Model on Speech Recognition

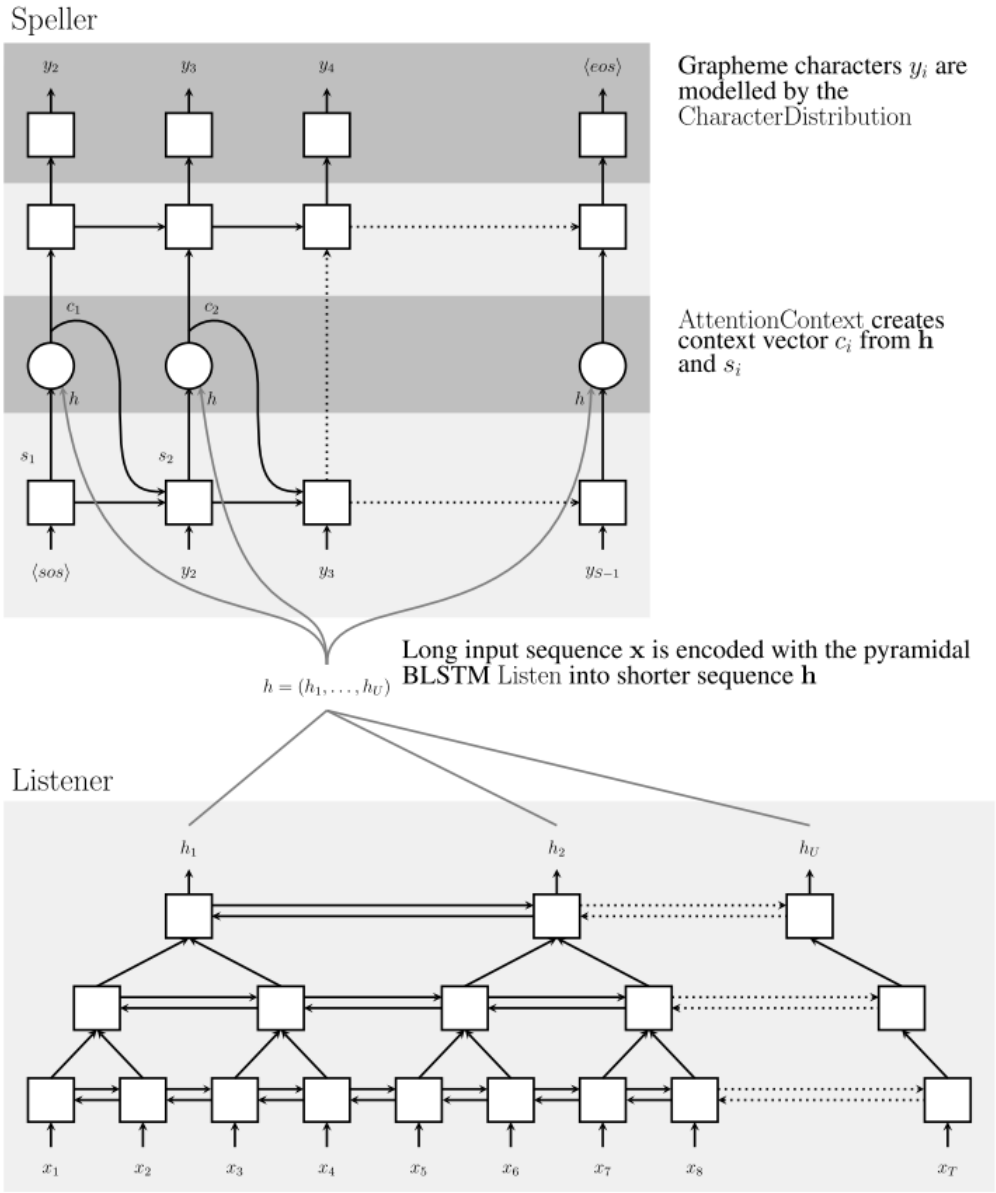
Todo List

1. Chiu, Chung-Cheng, et al. "State-of-the-art speech recognition with sequence-to-sequence models." *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018.
2. Zou, Wei, et al. "Comparable study of modeling units for end-to-end mandarin speech recognition." *2018 11th International Symposium on Chinese Spoken Language Processing (ISCSLP)*. IEEE, 2018.
3. Park, Daniel S., et al. "Specaugment: A simple data augmentation method for automatic speech recognition." *arXiv preprint arXiv:1904.08779* (2019).
4. Hannun, Awni, et al. "Deep speech: Scaling up end-to-end speech recognition." *arXiv preprint arXiv:1412.5567* (2014).
5. Amodei, Dario, et al. "Deep speech 2: End-to-end speech recognition in english and mandarin." *International conference on machine learning*. 2016.
6. Battenberg, Eric, et al. "Exploring neural transducers for end-to-end speech recognition." *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2017.

Listen, Attend and Spell

Notes

1. 模型架构：分成两个模块，一个是 Listen Encoder 模块，从语音时序序列中提取出高维特征，采用 pBLSTM (pyramid BLSTM) 的架构；另一个是 Attend and Spell 模块，从语音高维特征中输出单词，采用 Attention + LSTM 架构。架构图如下：



2. Listen Encoder 模块，使用 pBLSTM 的架构，每层在时间维度上减少一倍，带来的优点有两个：

   (1) 减少模型的复杂性；

   (2) 加快模型的拟合速度（作者发现直接用 BLSTM 的话，用一个月的时间训练都没有办法得到好的结果）；

   形式化的公式为：

$$h_i^j = \text{pBLSTM}(h_{i-1}^j, \left[h_{2i}^{j-1}, h_{2i+1}^{j-1}\right])$$

3. Attend and Spell 模块，该模块采用 2 层 LSTM 单元来记忆模型当前的状态 s (由模型上一次的状态、输出字符和上下文信息转化而来)，Attention 单元根据当前的状态 s 从特征 h 中分离出“当前模型关心的”上下文信息 c，最后 MLP 单元根据模型的状态 s 和上下文信息 c 输出最可能的字符 y。形式化的公式如下：

$$\begin{aligned} c_i &= \text{AttentionContext}(s_i, \mathbf{h}) \\ s_i &= \text{RNN}(s_{i-1}, y_{i-1}, c_{i-1}) \\ P(y_i|\mathbf{x}, y_{<i}) &= \text{CharacterDistribution}(s_i, c_i) \end{aligned}$$

   其中 Attention 单元在模型中的实现：将模型状态 s 和特征 h 分别经过两个不同的 MLP 模型，计算出一个标量能量 (Scalar Energy，相当于一个相关性系数) e，然后用 softmax 处理一下这个概率后，和原来的特征 h 加权生成上下文信息 c。形式化的公式如下：

$$e_{i,u} = \langle \phi(s_i), \psi(h_u) \rangle$$
$$\alpha_{i,u} = \frac{\exp(e_{i,u})}{\sum_u \exp(e_{i,u})}$$
$$c_i = \sum_u \alpha_{i,u} h_u$$

4. Learning. 模型的目标是，在给定 **全部** 语音信号和 **上文** 解码结果的情况下，模型输出正确字符的概率最大。形式化的公式如下：

$$\max_\theta \sum_i \log P(y_i | \mathbf{x}, y^*_{<i}; \theta)$$

在训练的时候，我们给的 y 都是 ground truth，但是解码的时候，模型不一定每个时间片都会产生正确的标签。虽然模型对于这种错误是具有宽容度，单训练的时候可以增加 **trick**：以 **10%** 的概率从前一个解码结果中挑选 (根据前一次的概率分布) 一个标签作为 ground truth 进行训练。形式化公式如下：

$$\tilde{y}_i \sim \text{CharacterDistribution}(s_i, c_i)$$
$$\max_\theta \sum_i \log P(y_i | \mathbf{x}, \tilde{y}_{<i}; \theta)$$

另外，作者发现预训练 (主要是预训练 Listen Encoder 部分) 对 LAS 模型没有作用。

5. Decoding & Rescoring. 解码的时候使用 Beam-Search 算法，目标是希望得到概率最大的字符串。形式化公式如下：

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} \log P(\mathbf{y}|\mathbf{x})$$

可以用语言模型对最后一轮 Beam-Search 的结果进行重打分，形式化公式如下：

$$s(\mathbf{y}|\mathbf{x}) = \frac{\log P(\mathbf{y}|\mathbf{x})}{|\mathbf{y}|_c} + \lambda \log P_{\text{LM}}(\mathbf{y})$$

增加解码结果的长度项 |y| 来**平衡产生长句、短句的权重**，另外语言模型的权重 lambda 可以通过验证集数据来确定。

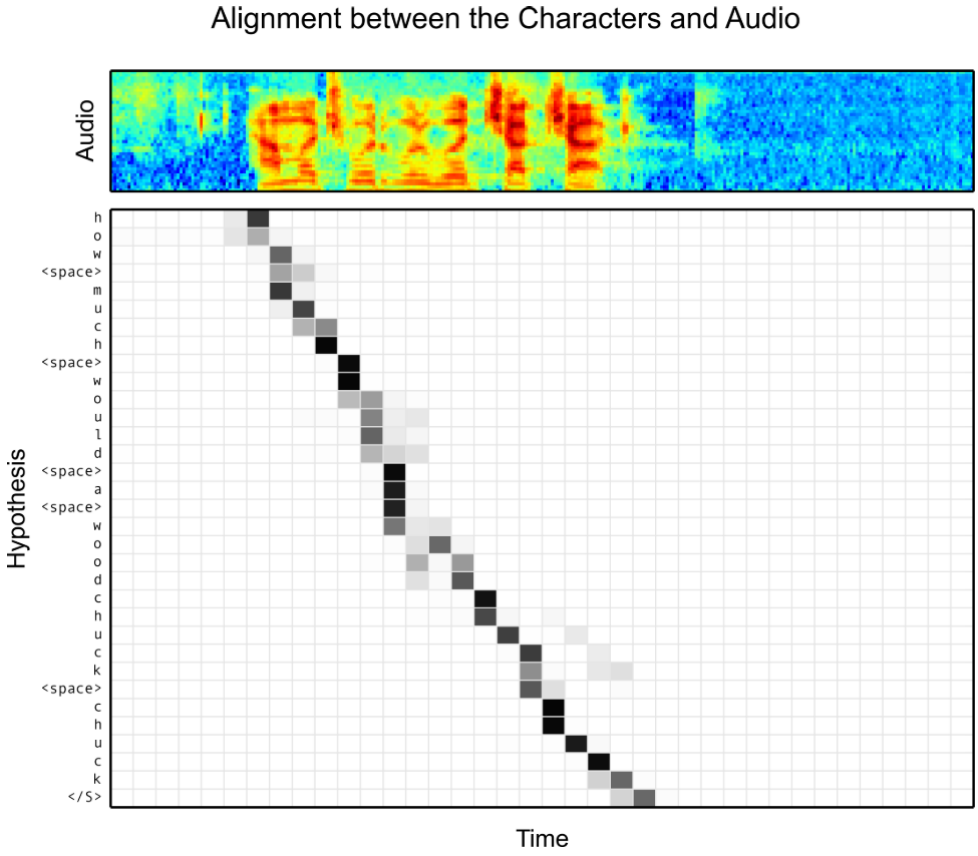
6. 实验结果：

(1) 使用 log-mel filter bank 特征

(2) 整体对比，LAS 刚出来的时候并打不过传统的 DNN-HMM 模型；

Model	Clean WER	Noisy WER
CLDNN-HMM [20]	8.0	8.9
LAS	16.2	19.0
LAS + LM Rescoring	12.6	14.7
LAS + Sampling	14.1	16.5
LAS + Sampling + LM Rescoring	10.3	12.0

(3) Attention 模块确实更加关注对应时间片段的特征；



(4) 模型对于较短的语句或者较长的语句效果都不是很好；

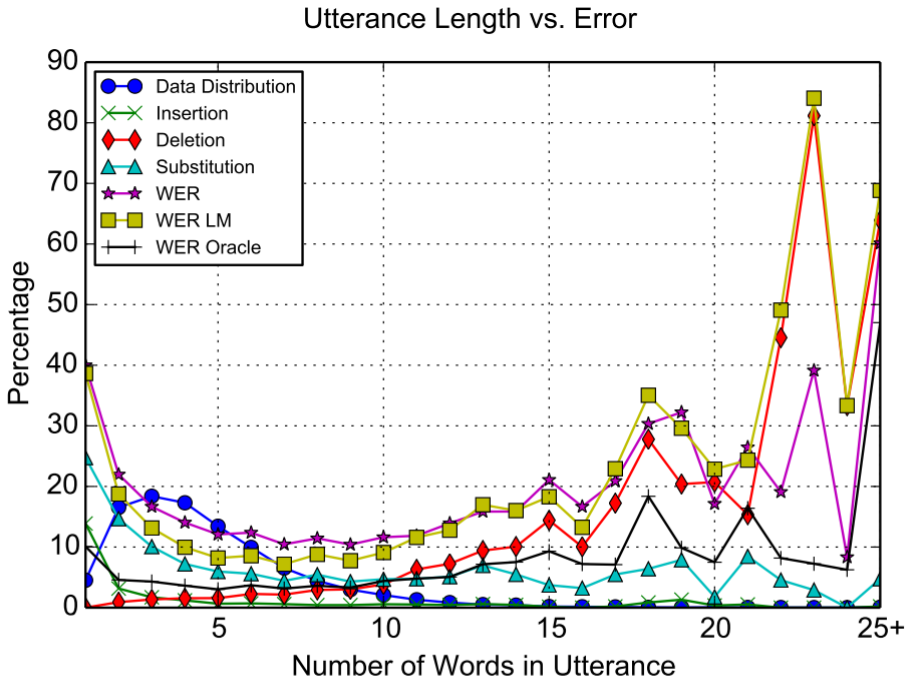


Figure 4: The correlation between error rates (insertion, deletion, substitution and WER) and the number of words in an utterance. The WER is reported without a dictionary or language model, with language model rescoring and the oracle WER for the clean Google voice search task. The data distribution with respect to the number of words in an utterance is overlaid in the figure. LAS performs poorly with short utterances despite an abundance of data. LAS also fails to generalize well on longer utterances when trained on a distribution of shorter utterances. Insertions and substitutions are the main sources of errors for short utterances, while deletions dominate the error for long utterances.

Shortcoming

1. 必须要得到整个语音后才能解码，限制了模型的流式处理能力；
2. Attention 机制需要消耗大量的计算量；
3. 输入长度对于模型的影响较大；

Links

- 论文链接：Listen, Attend and Spell
- LAS 模型缺点参考链接：LAS 语音识别框架发展简述
- Pytorch 实现：End-to-end-ASR-Pytorch (**暂未阅读代码**)

Lingvo: a modular and scalable framework for sequence-to-sequence modeling

谷歌开源的基于tensorflow的序列模型框架。

Notes

Links

- 论文链接：Lingvo: a modular and scalable framework for sequence-to-sequence modeling
- Github：Lingvo