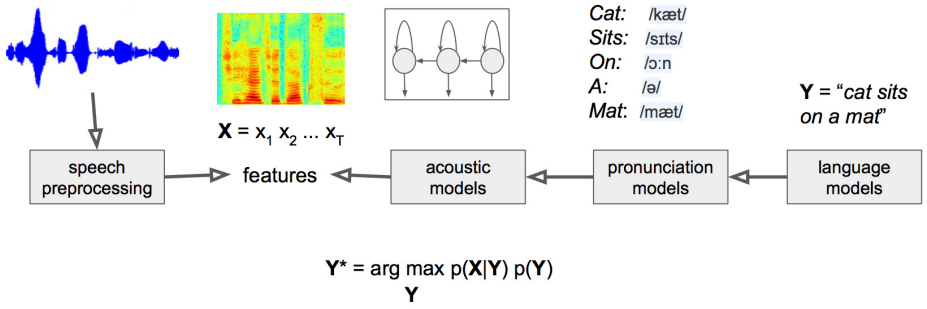
Lecture 12: End2End model for Speech Processing

2018 年 8 月 5 日

1 Automatic Speech Recognition（ASR）

1.1 传统语音识别

Building a statistic model of speech starting from text sequence to audio features.



建立 generative model，对每个词有一个语音信息

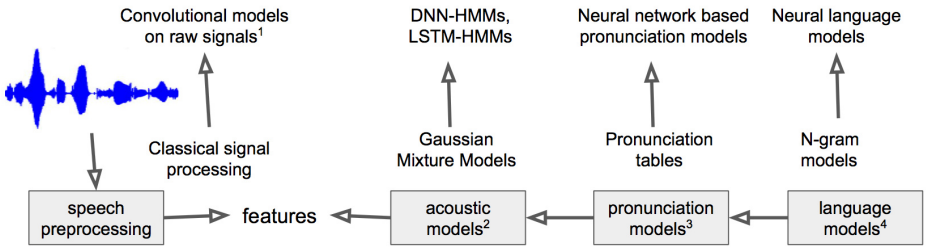
language model：一般是 N-gram model

pronunciation model：简单的 lookup tables，将文本转换成发音 token

acoustic model：将 pronounce token 送进来转换成语音，Gaussian mixture model 实现 spectrograms(声谱图).

通过 speech processing 将语音输入提取 audio features，given audio feature 通过某些 fancy search 算法寻找最有可能的 text。

1.2 Neural Network invasion



现在通过一系列 NN 替代原来的每个统计模型，发现效果比原来的都好。

- Language model: N-grams to Neural language models.
- Pronunciation model: predict token sequence from character sequences
- Acoustic Model：用 LSTM/DNN 合成音频

1.3 从生成模型到概率模型

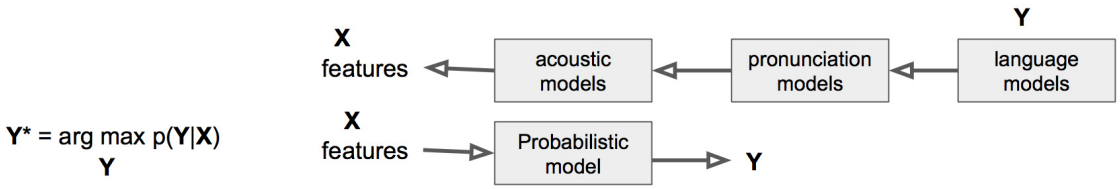
1.3.1 Generative model

同时对输入 X 和输出 Y 进行建模，得到联合分布 $P(X,Y)$, 然后根据 given X 预测 Y.

$$Y^* = \underset{Y}{argmax}\, P(Y|X)P(X)$$

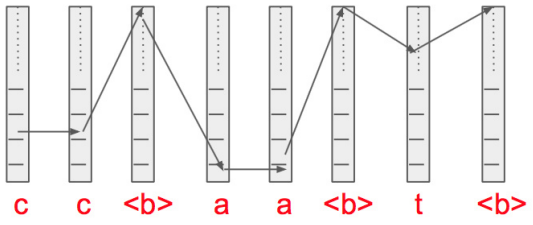
1.3.2 Discrimitive model

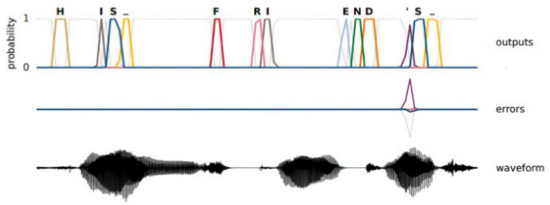
判别方法直接学习的是决策函数 $Y = f(X)$ 或者条件概率分布 $P(Y|X)$.



尽管使用 NN 准确性得到了提升, 然而各个子模型目标函数仍混乱, 各个部分 may not play well together. 现在使用一个大一统的 End2End 模型, 直接预测条件概率分布。

1.4 Still not satisfying



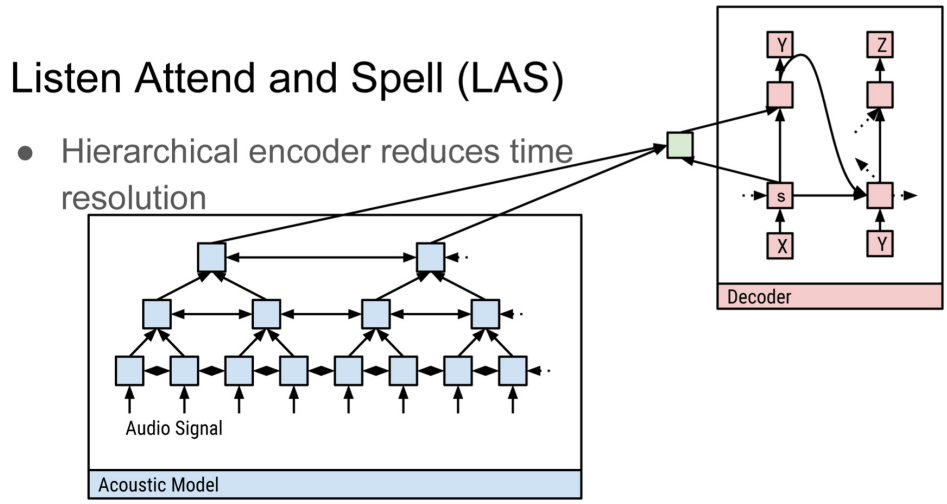


最简单的模型：底下音频做 BiLSTM 输入，上层 softmax 预测概率。

然而该模型误差很大（30 %），因为哪怕预测错一个字母，整个词就是错的。所以 Google 做了一个修补，在当前 End2End 模型外面套上一层 Language model，去修正个别错误的字母。然而这个模型就变得丑陋，并且不是完全的 End2End 了。

1.5 Listen Attend and Spell

Attention 机制的横空出世使得 Language model 可以成为整个 End2End 模型的有机组成部分。



1.5.1 Idea

这里将所有的音频信息输入 RNN，记录 hidden states，同时将文本信息作为另一个 sequence，类似于 NMT 中的 encoder-decoder 模型。

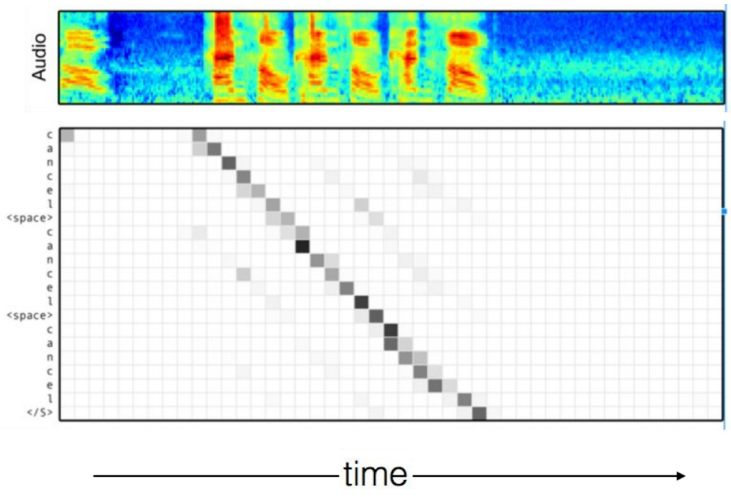
每个 time-step，根据当前 encoder 的隐藏状态 h_t 和 decoder 状态 s_{t-1} 计算该时间点 attended vector.

$$e_{ij} = a(s_{i-1}, h_j)$$

$$\alpha_{ij} = softmax_i(e_{ij})$$

在 decoder 的第 i 步，应该 attention on 哪些 encoder hidden states(如下图每一行所示)

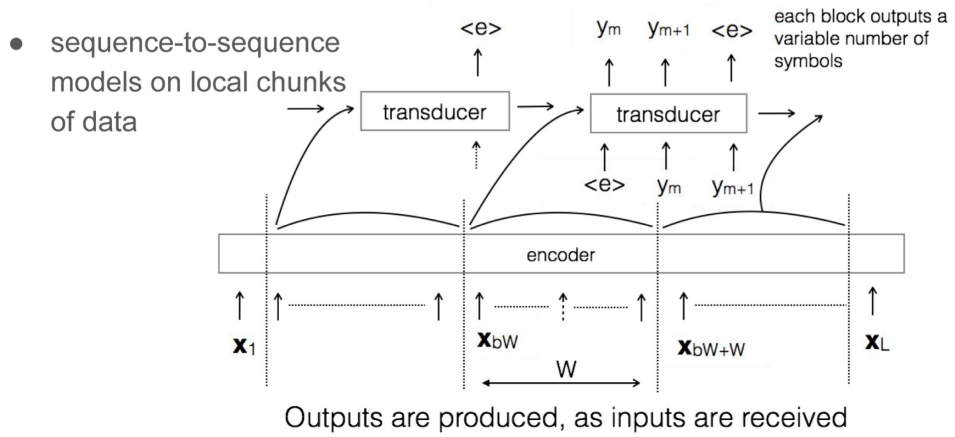
$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$



1.5.2 缺陷

1. Attention 不具备实时性，用户说完一整句话才能转换
2. Attention 大量使用 softmax，计算代价高
3. 句子长度越长，准确率越低（Attention 的全局性导致注意力分散）

2 Neural Transducer



将输入片段切分为一个个定长的 Chunk，每个 Chunck 对应输出任意长度个字符，最终由 Beam search 进行“分词”，找到最可能的路径。

A Neural Transducer - Training

- Correct gradient of log likehood:

$$\sum_{\tilde{y}\in\mathcal{Y}} p\left(\tilde{y}_{1\cdots(S+B))}|\mathbf{x}_{1\cdots L}, y_{1\cdots S}\right) \frac{d}{d\theta}\log p\left(\tilde{y}_{1\cdots(S+B)}|\mathbf{x}_{1\cdots L}\right)$$

- Viterbi-like training works well in practice:

$$\frac{d}{d\theta}\log p\left(\tilde{y}_{1\cdots(S+B)}|\mathbf{x}_{1\cdots L}\right)$$

(forced alignment with a DNN-HMM system works well too!)

$$\tilde{y}_{1\cdots(S+B)} = \arg\max_{\hat{y}_{1\cdots(S+B)}} p\left(\hat{y}_{1\cdots(S+B))}|\mathbf{x}_{1\cdots L}, y_{1\cdots S}\right)$$