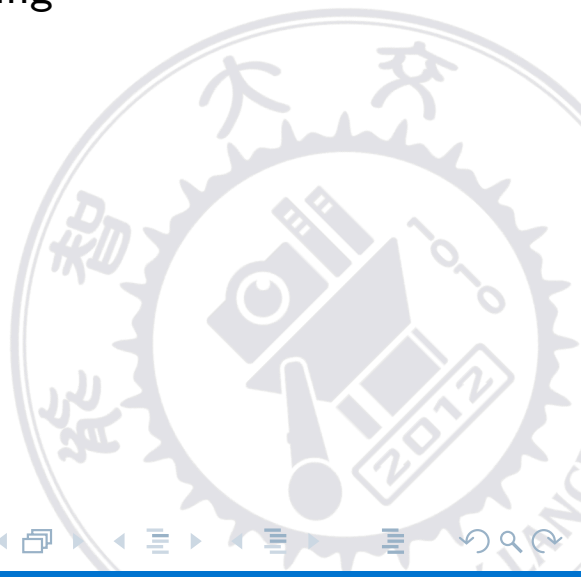
Lecture 5: 统计语音识别的基本概念

Basic Concept of Statistical Speech Recognition

Kai Yu and Yanmin Qian

Cross Media Language Intelligence Lab (X-LANCE)
Department of Computer Science & Engineering
Shanghai Jiao Tong University

Spring 2021



ASR: 自动语音识别

- 统计语音识别的基本概念和原理
- 特征提取
- 隐马尔科夫模型
- 统计语言模型
- 大词汇连续语音识别及搜索
- 深度神经网络模型及相关应用
- 端到端语音识别

ASR: 自动语音识别

参考教材:

《The HTK Book (for HTK version 3.4.1)》 Cambridge University, S. J. Young, etc.

《Foundamentals of Speech Recognition》 Lawrence Rabiner & Biing-Hwang Juang

《Spoken Language Processing》 Xuedong Huang, etc.

目录

自动语音识别是什么？

用模式匹配技术辨别一句话中的**字/词**（语音 $\rightarrow$ 转写文本）

语音识别的各种任务

典型的语音识别任务主要有如下几类:

- 孤立词 vs. 连续语音识别
- 有限词表（小词表或中等词表）vs. 大词表
- 限制语法网络 vs. 非限制开放网络
- 安静环境（干净信道）vs. 噪声环境（复杂信道）

目前语音识别的主要研究趋势和任务:

- 大词表及复杂语言领域
- 说话人无关
- 复杂多变的噪声信道和环境
- 小型化、本地化、芯片化

语音识别系统的相关配置

- **语音:**
  - 说话风格: 孤立的, 连接的, 连续的
  - 说话人集合: 单一的, 多个的, 说话人不相关的
- **环境:**
  - 安静（无噪声）, 低噪 (办公室), 强噪声 (汽车, 马路, 机场等.)
  - 麦克风/信道: 近讲麦克风, 远场环境, 电话信道等.
- **语言:**
  - 词表: 小词表, 中词表, 大词表
  - 语法: 上下文自由的语法, N 元语言模型 (statistical)
  - 语种: 中文, 英文, 阿拉伯文等.
- **交互:** 单一的 v.s. 多模态的
- **系统输出:** 单一候选路径, 多候选路径 (N-best list or lattices)
- **输出单元:** 词, 字, 子词 (音素, 音节等.)

语音识别系统

什么是一个“好”的语音识别系统？

- 定量的指标: **低的词或字错误率** w.r.t. 人工标注
- 定性的指标: **用户满意度**, i.e. 系统可用性

语音识别系统的设计很大程度依赖于具体的任务

- 隐式/显示的确认
- 是否允许校正 (多候选)
- 置信度
- 词表/拼写等的扩展

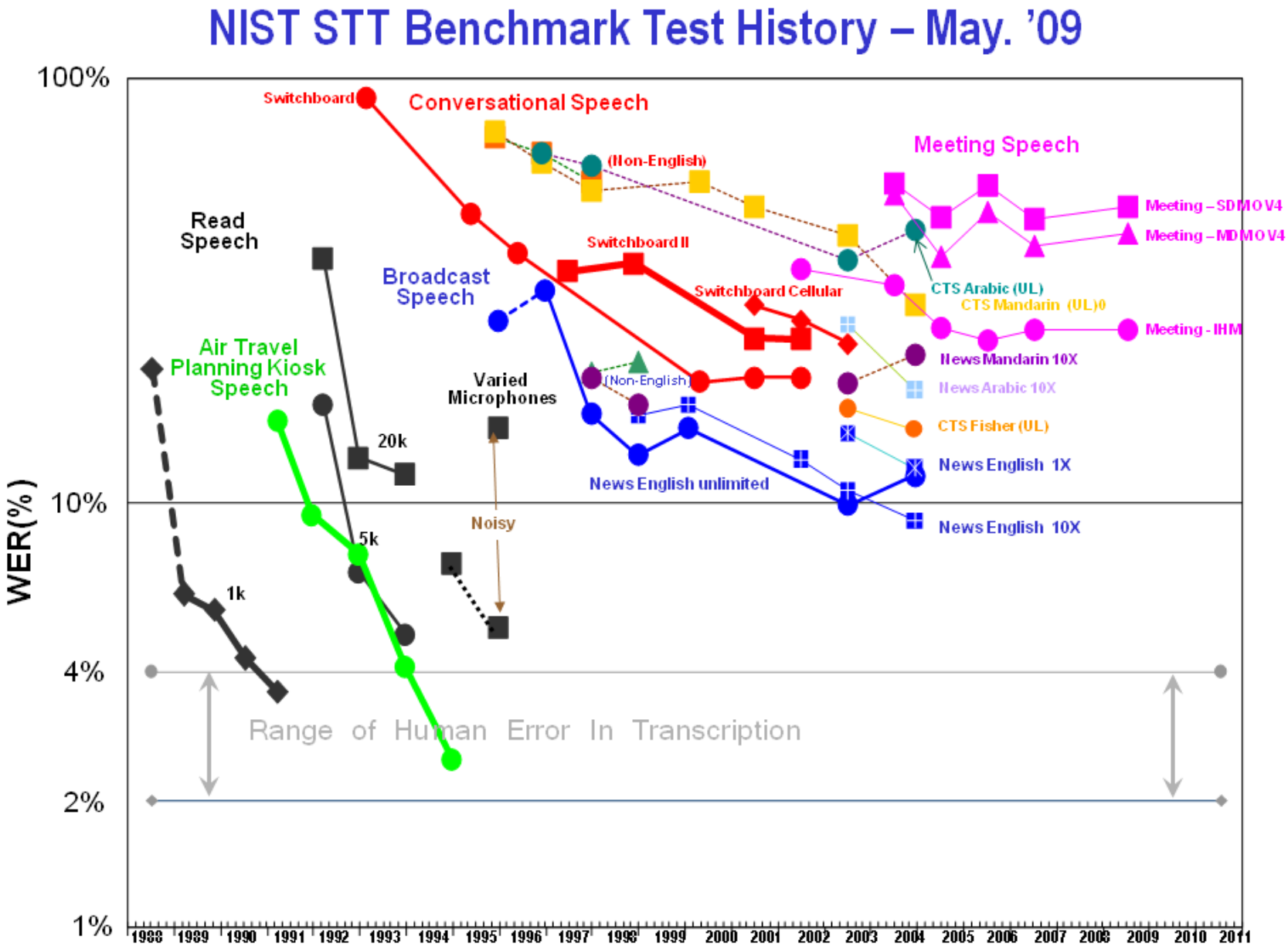
语音识别的历史

- **Early 1970s:** Dynamic Time Warping (DTW) 动态时间规整方法问世，用于处理时间序列的多变性，可用作语音谱变化的距离度量。
- **Mid-Late 1970s:** Hidden Markov Models (HMMs) 隐马尔科夫模型 — 利用统计模型来对语音谱变化进行建模。IBM 的 Jelinek 和 CMU 的 Baker 将该模型用于离散语音。
- **Mid-Late 1970s:** ARPA Speech Understanding program：构建 1000 词表大小的说话人相关语音识别系统。当时最好系统采用 HMM-like 方法。
- **Mid 1980s:** 基于 HMM 的统计建模方法成为最主要语音识别技术
- **Mid-Late 1980s:** ARPA Resource Management (RM) 任务：构建 1000 词表大小的说话人无关语音识别系统。基于 HMM 的语音识别方法得到进一步发展。
- **Late 1980s:** 第一个基于 HMM 的商用听写机系统问世 — Dragon (说话人相关的孤立词语音识别)。基于大词表的识别任务性能也逐步改进，同时基于神经网络的识别技术开始得到关注。HMM/NN 混合模型发展起来，采用非线性判别式函数进行概率估计。

语音识别的历史

- **Early-Mid 1990s:** ARPA Wall Street Journal (newspaper dictation)：5k → 非限制词表大小。HMM 方法的进一步高速发展，开始使用更复杂的建模和更大的数据集，HTK 出现。
- **Mid-Late 1990s:** 研究开始关注大词汇连续语音识别（LVCSR）(e.g. broadcast news, conversational telephone speech)。基于说话人和环境自适应的相关技术提出。DARPA 项目引领技术前沿。
- **1994-1998:** 从特定的应用（如医学听写）到桌面 PC 系统上的大词汇连续语音识别。需要注册（enrolment）数据用于说话人自适应。桌面听写开始兴起 (e.g. Dragon, IBM, LHS, Philips)。
- **1996 onwards:** 更精细的电话语音识别 + 简单对话管理 (stock-quotes, travel info, etc.)。一些语音公司开始快速发展，如 Nuance 和 SpeechWorks。
- **2000s:** 用于 LVCSR 系统鉴别性训练（discriminative training）算法发展起来。DARPA 的 EARS 及 GALE 项目成为集大成者。
- **2010s:** 深度学习（Deep learning）极大降低 LVCSR 系统的错误率。

语音识别的历史


NIST STT Benchmark Test History – May. '09
100%
10%
4%
2%
1%
WER(%)
Switchboard
Conversational Speech
(Non-English)
Meeting Speech
Meeting – SDMOV4
Meeting – MDMOV4
Read Speech
Switchboard II
Broadcast Speech
Switchboard Cellular
CTS Arabic (UL)
CTS Mandarin (UL)0
Meeting - IHM
Air Travel Planning Kiosk Speech
Varied Microphones
(Non-English)
News Mandarin 10X
News Arabic 10X
CTS Fisher (UL)
20k
News English 1X
News English unlimited
Noisy
5k
News English 10X
1k
Range of Human Error In Transcription
1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011

统计语音识别

ASR 的目标：找到最可能生成语句 O 的句子（词序列）W：

$$\hat{\mathbf{W}} = \arg\max_{\mathbf{W}} P(\mathbf{W}|\mathbf{O}) = \arg\max_{\mathbf{W}} p(\mathbf{O}|\mathbf{W})P(\mathbf{W})$$

主要部分包括：

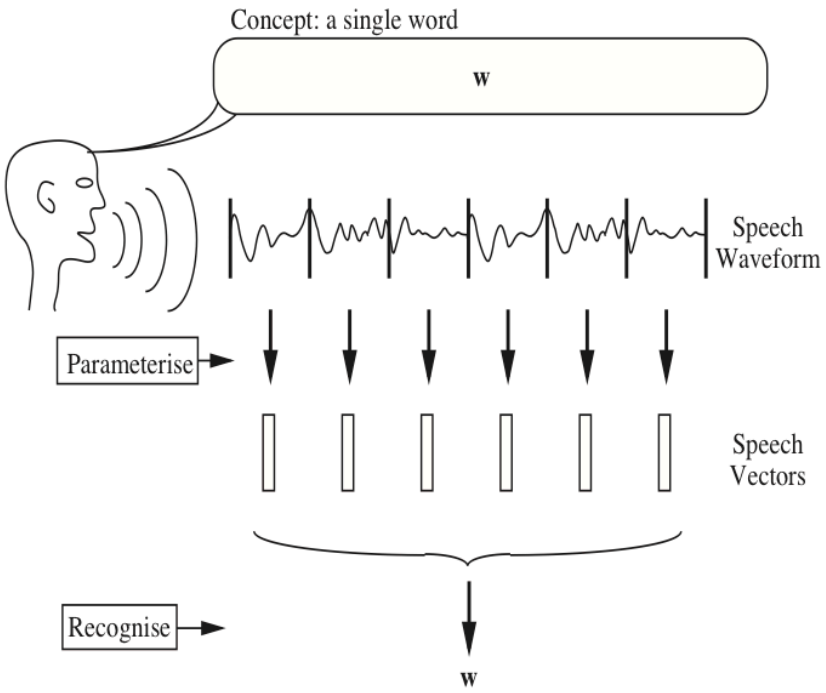
- 声学模型 $p(\mathbf{O}|\mathbf{W})$
- 语言模型 $P(\mathbf{W})$

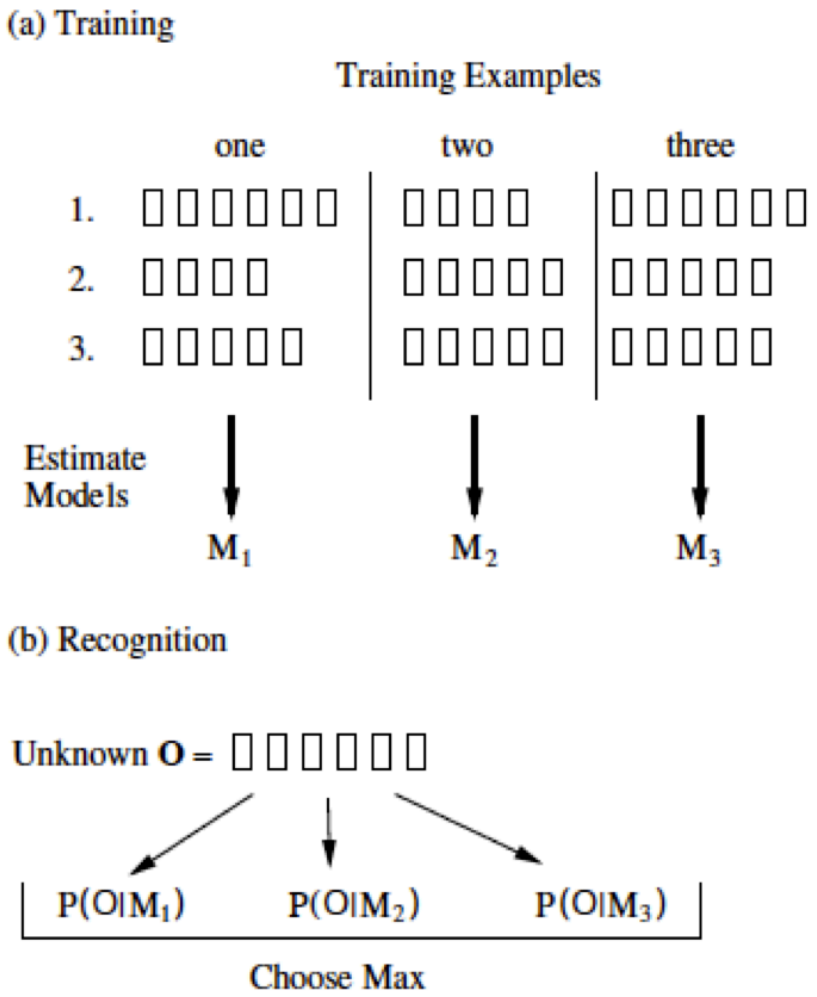
ASR 的相关研究

- **训练**：获得声学模型和语言模型最合适的模型结构和模型参数
- **解码**：识别得到输入音频的最可能词序列

统计语音识别

孤立词识别图示





统计语音识别

大词汇连续语音识别系统架构

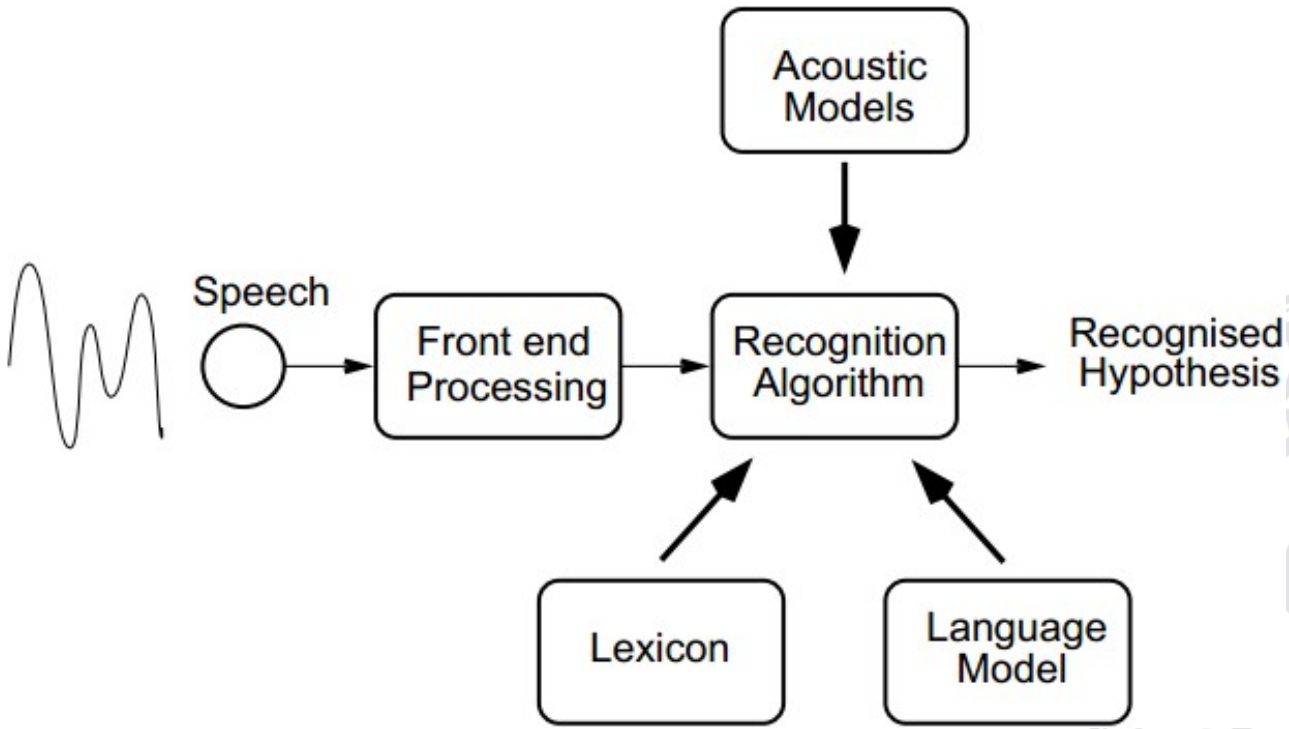
$$\hat{\mathbf{W}} = \arg\max_{\mathbf{W}} p(\mathbf{A}|\mathbf{O})p(\mathbf{O}|\mathbf{L})P(\mathbf{L}|\mathbf{W})P(\mathbf{W})$$

W: 词序列

L: 声学建模单元序列

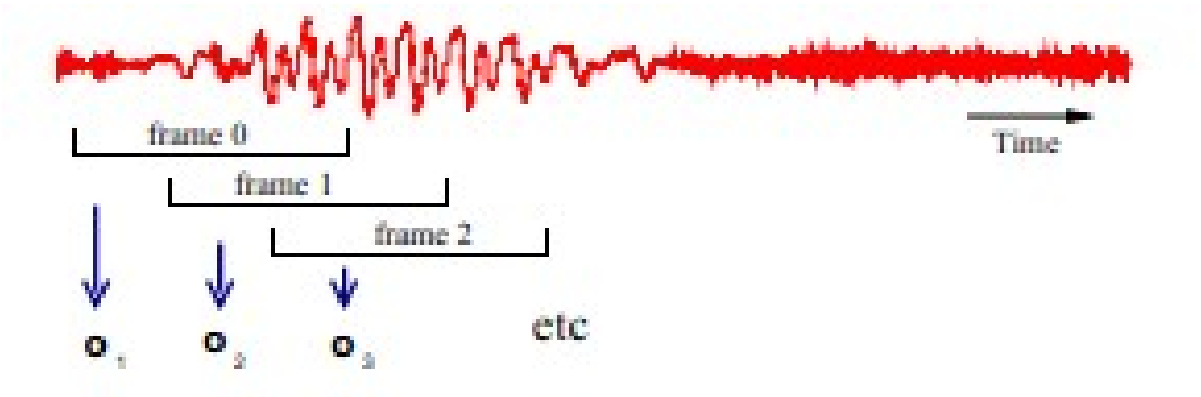
O: 音频样本序列

A: 声学特征序列



前端处理 — 特征提取 $p(\mathbf{A}|\mathbf{O})$

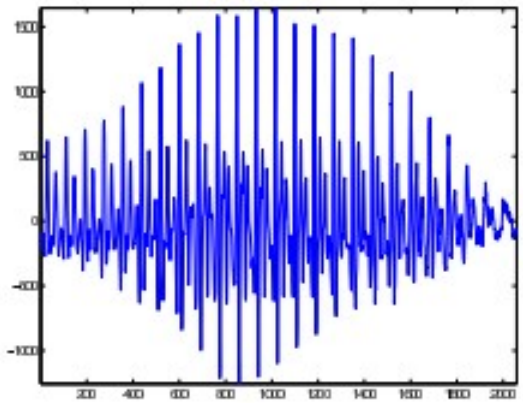
原始语音通过信号处理的方法转换成**特征向量**的序列。

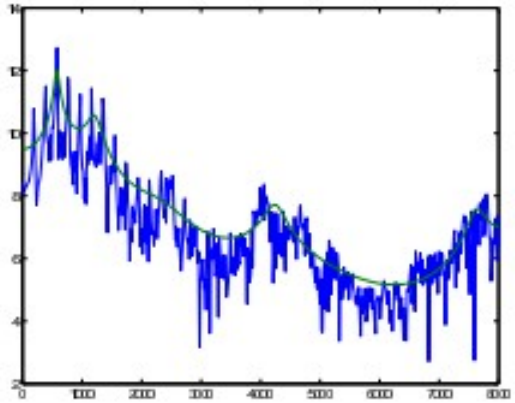


- 特征提取是一个**确定**的过程，i.e. $p(\mathbf{A}|\mathbf{O}) = \delta(\mathbf{A}, \hat{\mathbf{A}}(\mathbf{O}))$
- 降低信息率，但是保留有用信息
- 去除噪声或者其他的无关信息
  - 识别元音：最低的两个共振峰
  - 识别性别：音调（pitch）或者基音周期频率

前端处理 — 特征提取 $p(\mathbf{A}|\mathbf{O})$

语音识别中常见的声学特征





我们往往用短时谱信息来描述语音信号。一些有用的短时谱分析分析技术用于提取有用的特征，包括:

- 线性预测系数 (LPC)
- 滤波器组系数（Fbank）
- 梅尔频率倒谱系数 (MFCC)
- etc.

给定信号 $\mathbf{x} = [x_1, \cdots, x_T]^\top$，通过 n 个之前时刻样本的 n 阶带权线性插值来线性预测 t 时刻的样本值:

$$\hat{x}_t = \sum_{i=1}^{n} a_i x_{t-i} \quad \Rightarrow \quad \hat{\mathbf{x}} = \mathbf{M}\mathbf{a}$$

其中

$$\mathbf{M} = \begin{bmatrix} x_0 & x_{-1} & x_{-2} & \cdots & x_{-n+1} \\ x_1 & x_0 & x_{-1} & \cdots & x_{-n+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{T-1} & x_{T-2} & x_{T-3} & \cdots & x_{-n+T} \end{bmatrix} \quad \mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}$$

其中 $a_i, 1 \le i \le n$ 被称为**线性预测系数 (LPC)**，M 被称为托普利兹矩阵。

找到线性预测系数：

$$\mathcal{L}_{\texttt{mse}} = \frac{1}{T}\sum_{t=1}^{T}(\hat{x}_t - x_t)^2 = \frac{1}{T}(\hat{\mathbf{x}} - \mathbf{x})^\top(\hat{\mathbf{x}} - \mathbf{x})$$

其中有

$$\hat{x}_t = \sum_{i=1}^{n} a_i x_{t-i} \quad 1 \le t \le T$$

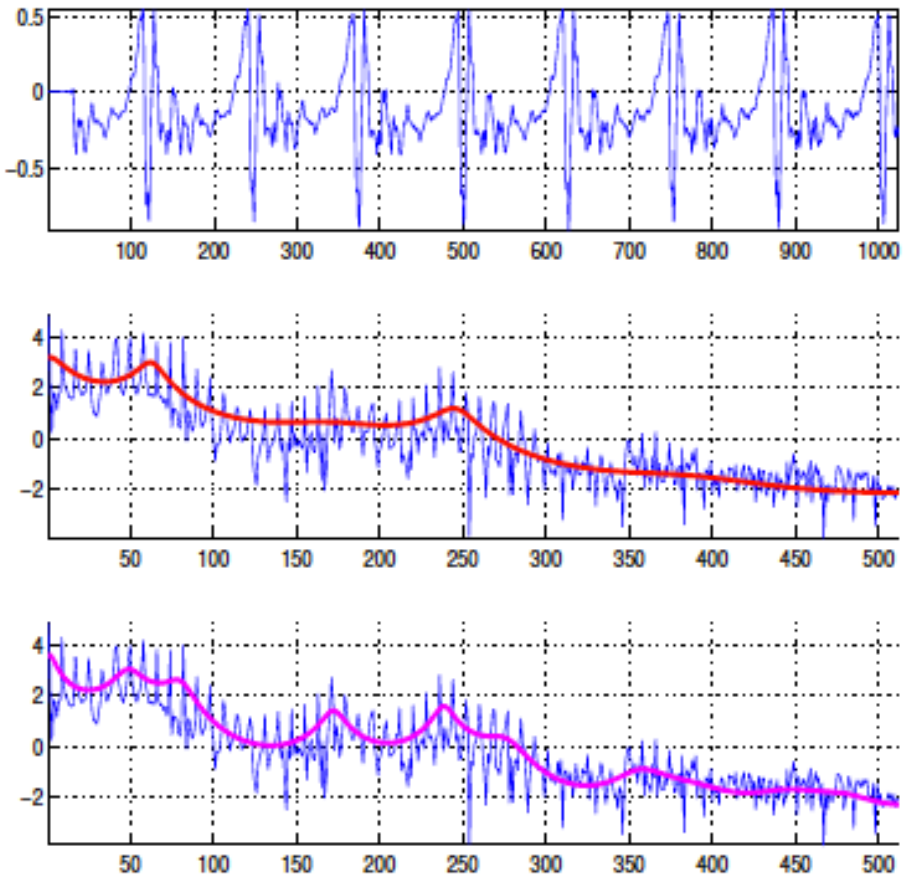
其中 $a_i, 1 \le i \le n$ **是线性预测系数**。

通常会直接用线性方程解法吗？**事实上不是**。

$$\mathbf{a} = (\mathbf{M}\mathbf{M}^\top)^{-1}\mathbf{M}^\top\mathbf{x}$$

Levinson-Durbin 算法是一个自相关算法，它可以利用托普利兹矩阵 M 的相关特性进行快速计算。

从 LPC 得到的谱包络



- TOP: 声音"aa" 的语音信号波形
- MIDDLE & BOTTOM: 对数标度下的语音谱幅值
- 用一条红色的平滑曲线画出的谱包络
  - MIDDLE: 10-order LP
  - BOTTOM: 25-order LP
- LPC 阶数越高，它对谱幅值的刻画越精确
- 谱包络的峰值可以用来确定共振峰的位置

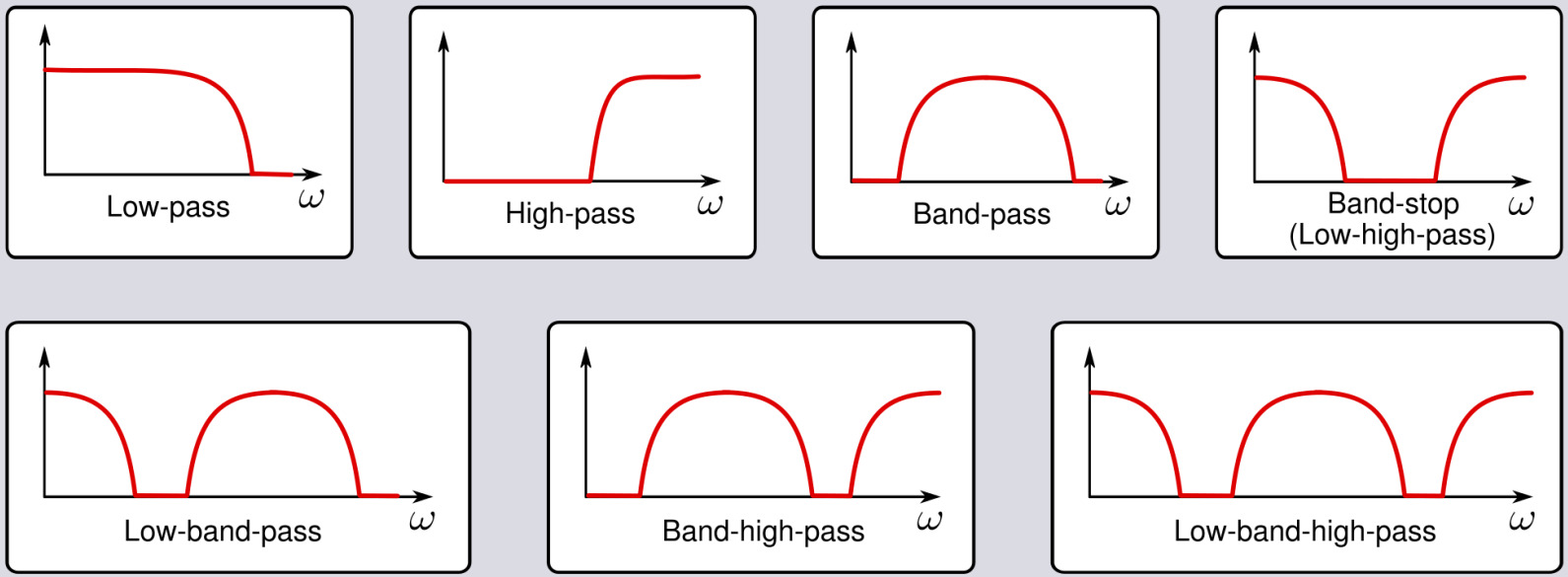
前端处理 — 特征提取 $p(\mathbf{A}|\mathbf{O})$

Filter 滤波器

滤波器 一般是一个设备或者处理过程，它可以从原始信号中将那一些不想要的成分或者特征去除

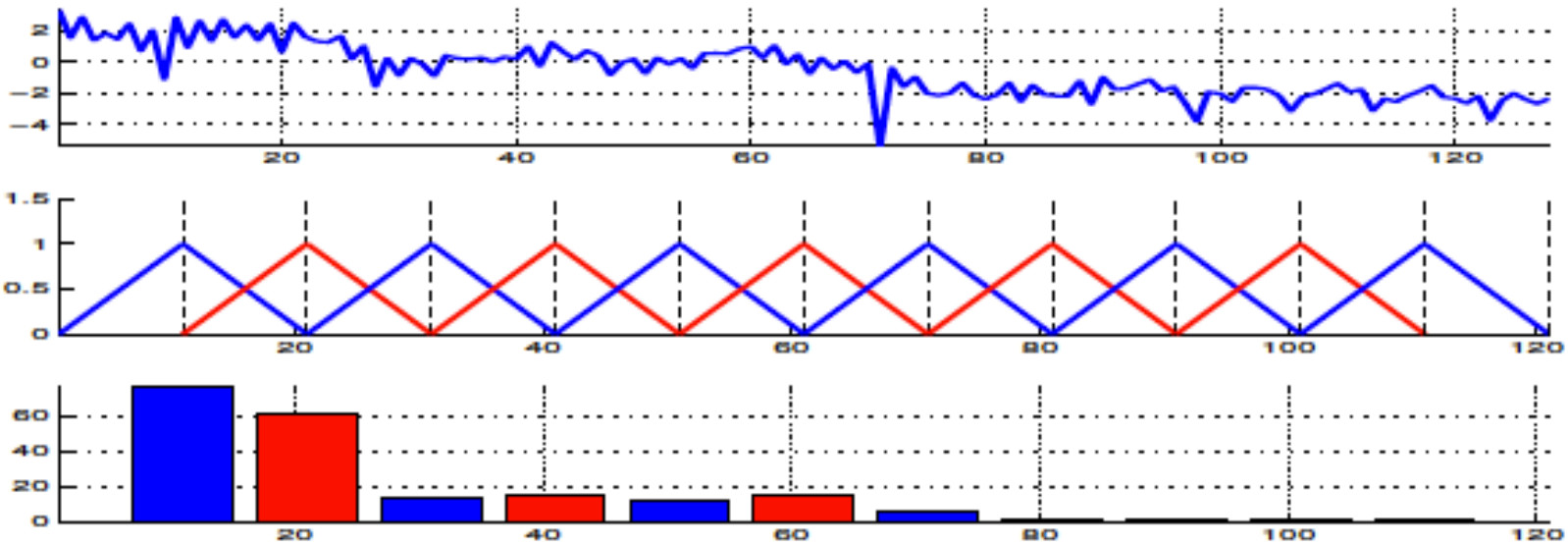
滤波操作一般情况下作用在频率域上：

$$Y(\omega) = H(\omega)X(\omega)$$



前端处理 — 特征提取 $p(\mathbf{A}|\mathbf{O})$

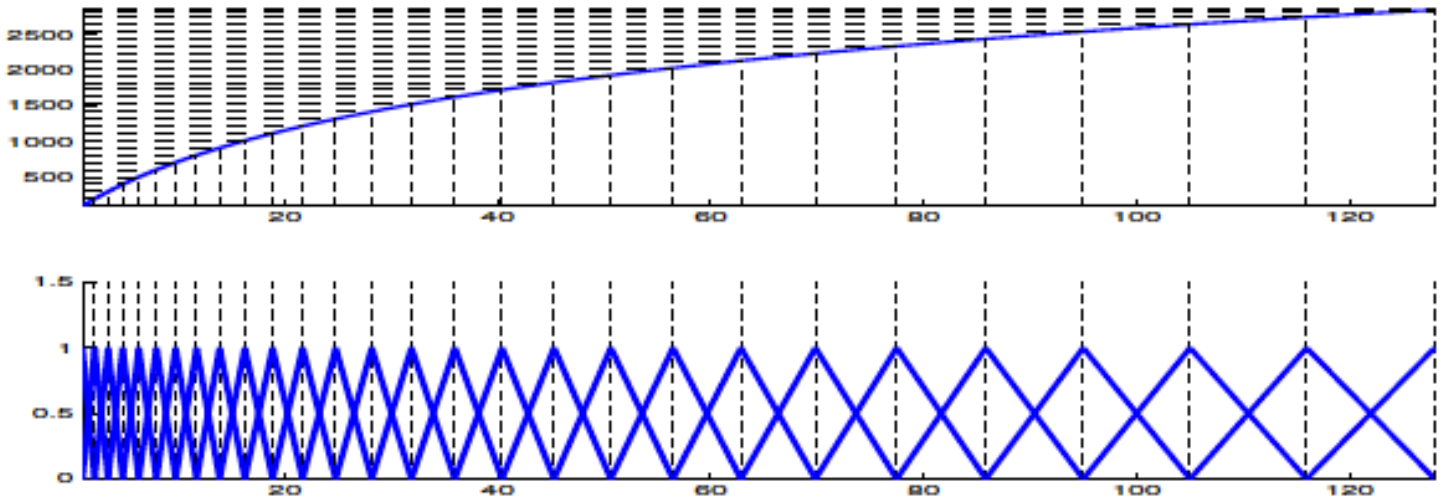
Filter Bank 系数（FBank）



- 通过短时傅里叶变换得到的幅值谱包含太多的信息
- **Filter bank** 是一系列带通滤波器 (一般用三角窗滤波器)
- 每一个带通滤波器输出一个 FBank 系数，它等于此带通滤波器内的信号加权和

前端处理 — 特征提取 $p(\mathbf{A}|\mathbf{O})$

梅尔域（Mel Scale）



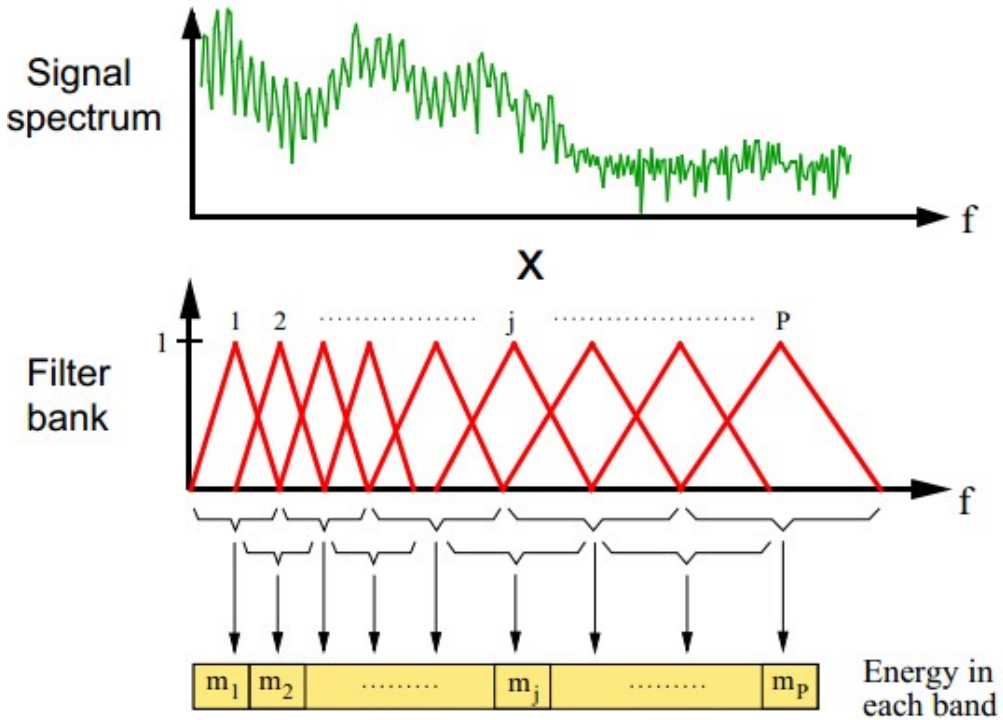
Mel scale 是一个基音感知域，它通过听音者可以区分两种纯音频率的差距来作为标度。Mel 域和线性频域之间通过一个非线性的映射函数可以相互转换：

$$\texttt{Mel}(f) = 2595 \log_{10}(1 + \frac{f}{700})$$

注意：在 Mel 域中，低频部分具有更高的分辨率。

前端处理 — 特征提取 $p(\mathbf{A}|\mathbf{O})$

梅尔域的 FBank 系数



$$m_i = \sum_{k=f_i}^{F_i} s(k) T_i(k)$$

其中 m_i 是第 i^{th} 个 FBank 系数，f_i 和 F_i 分别是三角滤波器的开始和结束频率，$s(k)$ 是在频点 k 的谱的能量 (有时为谱的幅值)，$T_i(k)$ 是三角滤波器的值。

MFCC 在许多语音处理的领域被广泛使用。它们从 Mel 域的 FBank 系数衍生得到：

1. 取对数 **logarithm** 得到 N 个对数域的 FBank 系数
2. 计算倒谱 **Cepstral** 系数，使用 **Discrete Cosine Transform (DCT)** 变换

$$c_n = \sqrt{\frac{2}{N_{\texttt{fb}}}} \sum_{j=1}^{N_{\texttt{fb}}} \log(m_j) \cos\left(\frac{\pi n}{N_{\texttt{fb}}}(j-0.5)\right) \quad n = 1, 2, \cdots, N_{\texttt{mfcc}}$$

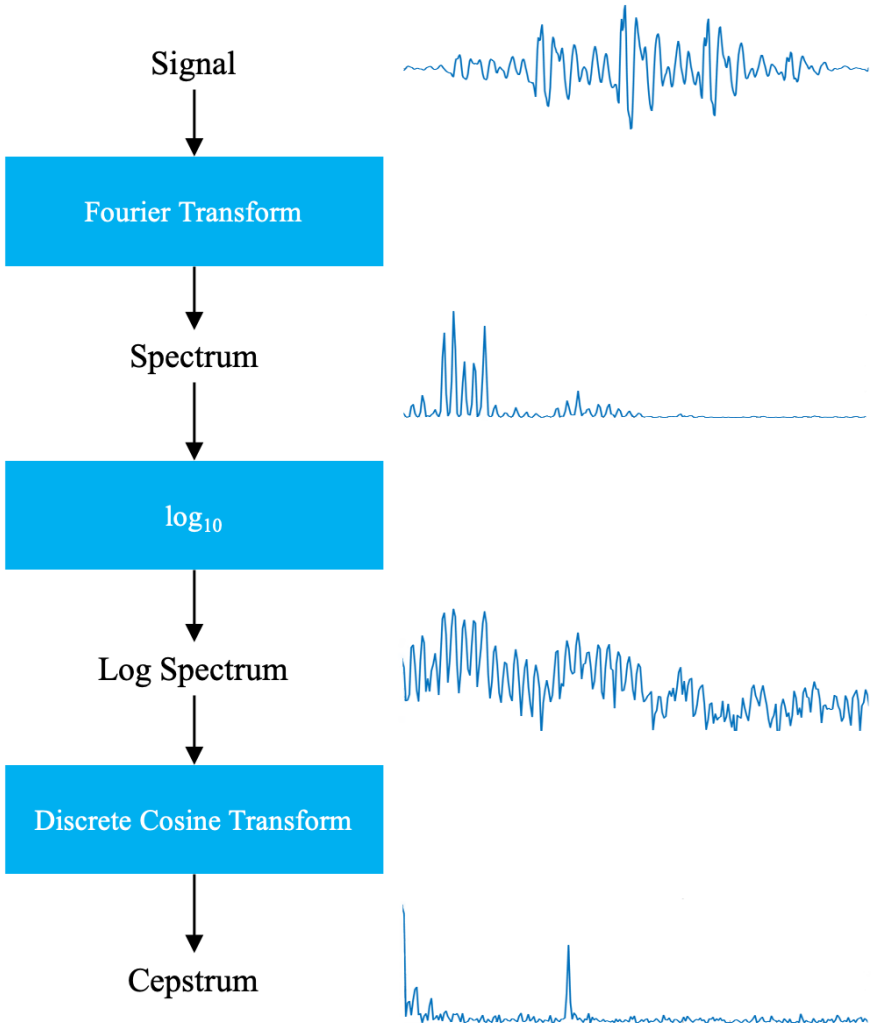
其中 c_n 是第 n^{th} 个 MFCC 系数，m_j 是第 j^{th} 个 mel 域的 FBank 系数，$N_{\texttt{fb}}$ 和 $N_{\texttt{mfcc}}$ 分别是 FBank 系数和最终的 MFCC 系数的数量 (通常情况下 $N_{\texttt{mfcc}} = 12$，$N_{\texttt{fb}}$ 变化范围是 20-30，甚至 40).

从频谱 (Spectrum) 到倒谱 (Cepstrum)

语音短时谱能量没有直接用来作为特征参数，因为

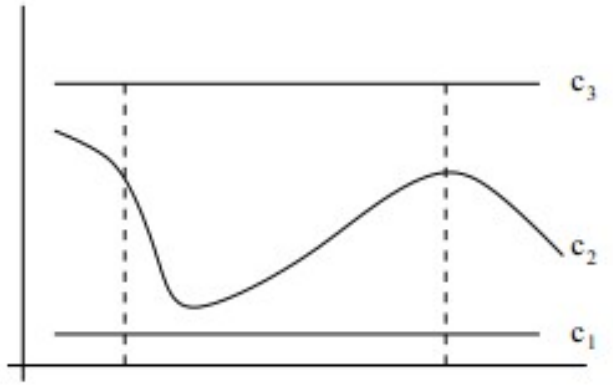
- 语音的能量谱不符合高斯分布
- 所有系数对响度很敏感
- 各个系数之间都高度相关

Discrete Cosine Transform (DCT) 能够有效地去除不同系数之间的相关性。



概念

MFCC 特征很好地描述了即时语音信号谱（静态），但是不能描述信号的动态特性。



这个不足可以通过在特征向量中添加静态系数的差分特征来得到改善。

$$\mathbf{o} = \begin{bmatrix} \mathbf{c} \\ \Delta\mathbf{c} \\ \Delta\Delta\mathbf{c} \end{bmatrix}$$

计算

- 简单的差分系数可以计算如下：

$$\Delta_n = \frac{\mathbf{c}_{n+\delta} - \mathbf{c}_{n-\delta}}{2\delta}$$

- 更鲁棒的估计是通过一系列语音帧的最佳线性回归系数来得到：(这里是 $2\sigma + 1$)

$$\Delta_n = \frac{\sum_{i=1}^{\delta} i(\mathbf{c}_{n+i} - \mathbf{c}_{n-i})}{2\sum_{i=1}^{\delta} i^2}$$

- 更高阶的差分系数可以通过以上过程的迭代形式来获得：

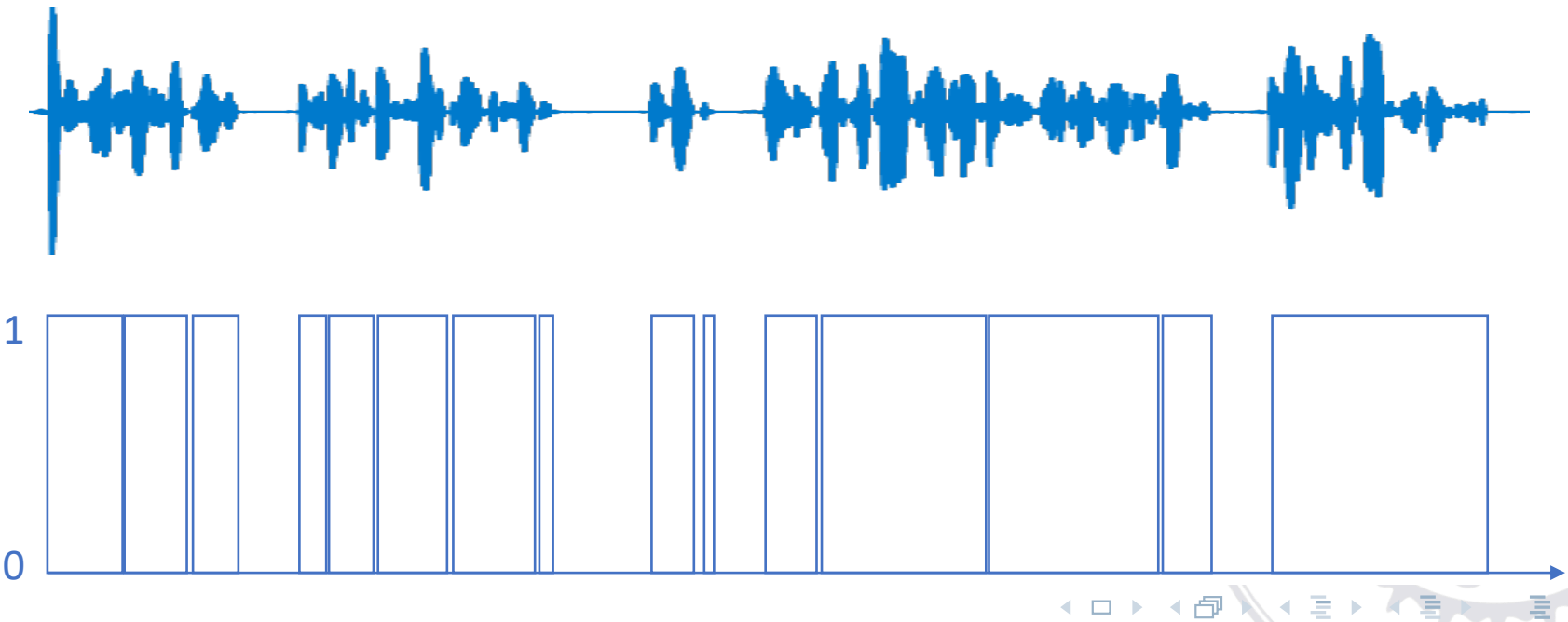
$$\Delta\Delta_n = \frac{\sum_{i=1}^{\delta} i(\Delta_{n+i} - \Delta_{n-i})}{2\sum_{i=1}^{\delta} i^2}$$

- 通常情况下会使用 2^{nd} or 3^{rd} 系数

前端处理 — 语音端点检测（VAD）

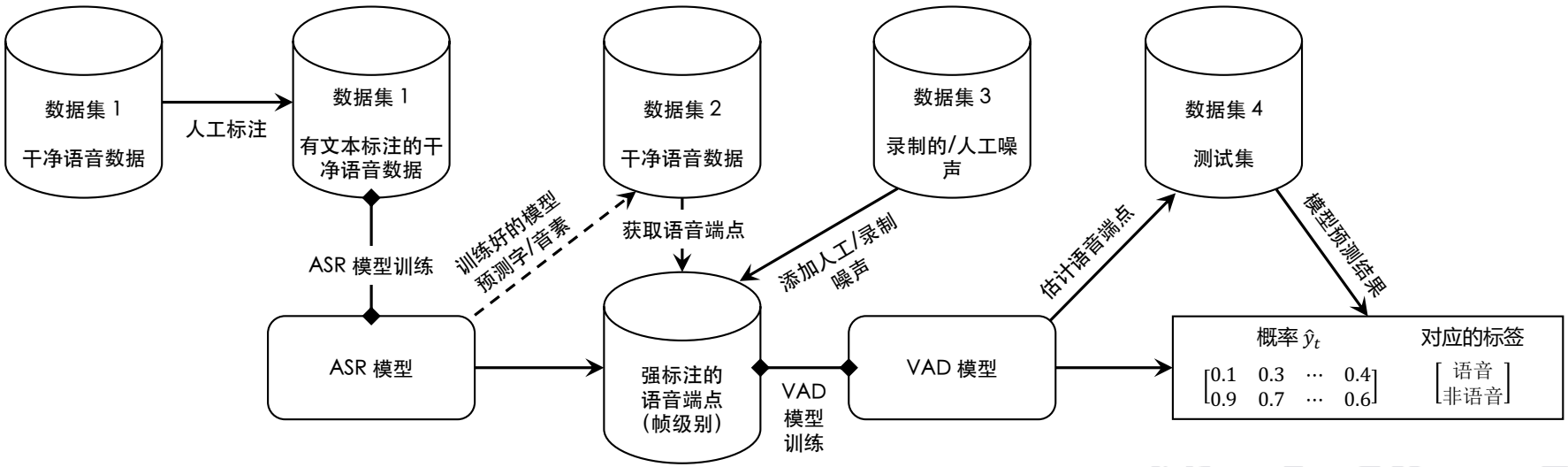
语音端点检测 — Voice Activity Detection (VAD)

- 主要任务：检测一段音频中是否存在人类的语音
- 作用：
  - 减少不必要的语音处理，节省运算开销和功耗
  - 也能提升后续语音处理的性能（减少了不必要的非语音噪声输入）



前端处理 — 语音端点检测（VAD）

传统的有监督 VAD 模型训练流程



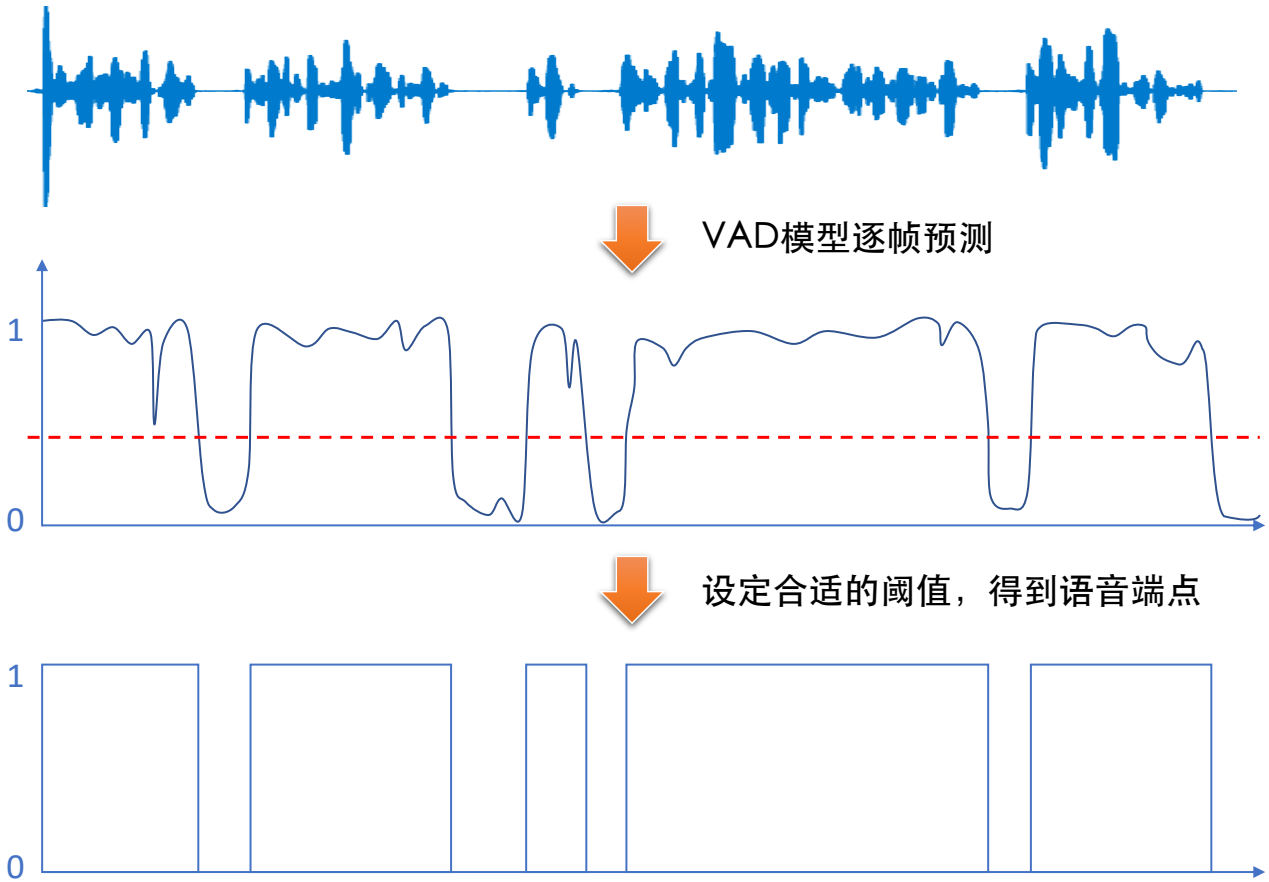
前端处理 — 语音端点检测（VAD）

传统的有监督 VAD 模型训练流程

- 1. 采集干净的单人语音数据，尤其是音频质量相差不大的数据。确保数据中仅包含固定种类的一种/多种语言。
- 2. 手动对这些数据进行文本标注。
- 3. 在这些数据上训练一个 ASR 模型。
- 4. 用训练好的 ASR 模型来预测新的（通常干净）数据集中的音频是否存在语音，以得到帧级别的强标注。
  - ASR 模型可以提供音素级别的对齐，进而转换为二值的语音指示（有语音的帧设为 1，静音段设为 0）
- 5. 在新数据集上用前一步得到的强标注训练一个 VAD 模型（DNN/CNN/RNN 等）。

前端处理 — 语音端点检测（VAD）

VAD 模型测试阶段



Q: 怎么评估 VAD 的性能？

分类/检测问题中的评估指标

- 准确率（Accuracy）
- 精确率/查准率（Precision）和召回率/查全率（Recall）
- F-指标（macro F1 score, micro F1 score, etc.）
- 误识率（False Acceptance Rate, FAR）和
  误拒率（False Rejection Rate, FRR）
- ROC 曲线（Receiver Operator Characteristic Curve）、
  ROC 曲线下面积（Area Under the ROC Curve, AUC）和
  等错误率（Equal Error Rate, EER）

分类/检测问题中的评估指标

基本概念

正样本（positives）和负样本（negatives）

- 对于二类问题，属于类别 1 的样本是正样本，属于类别 0 的样本是负样本；
- 对于多类问题，如果采用 one-vs.-rest 策略，则对于每一个类别 i 来说，属于该类别的样本为正样本，不属于该类别的样本为负样本。

		实际类别：正样本	实际类别：负样本
预测类别	预测为正样本	True Positive (TP)	False Positive (FP)
预测类别	预测为负样本	False Negative (FN)	True Negative (TN)

分类/检测问题中的评估指标

准确率 (Accuracy)

准确率：在所有样本中，预测正确的正样本和负样本所占比例

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}$$

预测类别 \ 实际类别	正样本	负样本
预测为正样本	True Positive (TP)	False Positive (FP)
预测为负样本	False Negative (FN)	True Negative (TN)

Q: 仅用准确率评估 VAD 的性能有什么问题？

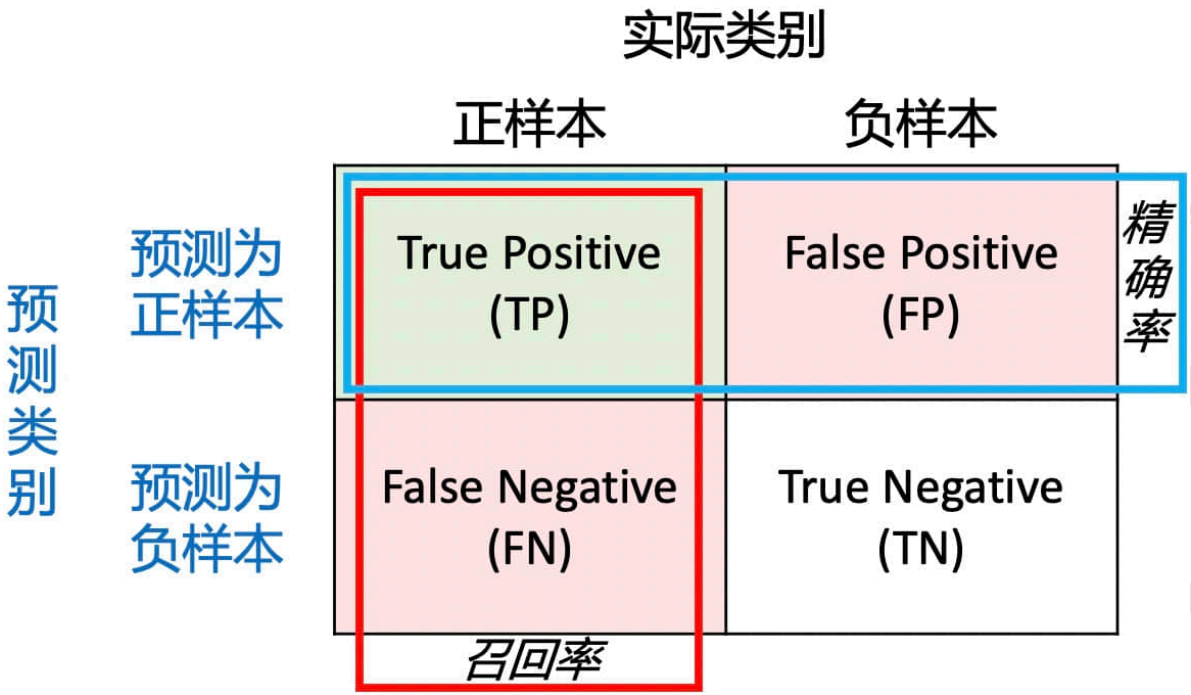
分类/检测问题中的评估指标

精确率/查准率（Precision）和召回率/查全率（Recall）

精确率/查准率 (Precision)：在预测为正样本的样本中，预测正确的比例

召回率/查全率 (Recall)/TPR (True Positive Rate)：在所有正样本中，被预测正确的比例

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$



分类/检测问题中的评估指标

F-指标

对于二类检测问题：

- F_β 值：精确率和召回率的（加权）调和均值

$$F_\beta = (1+\beta^2)\frac{\text{Precision} \times \text{Recall}}{\beta^2 \times \text{Precision} + \text{Recall}}$$

其中 β 的取值可根据精准率和召回率在评估时的相对重要性来决定，通常取 1：

$$F_1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2 \times \text{TP}}{2 \times \text{TP} + \text{FP} + \text{FN}}$$

对于 N 类检测问题（$N > 2$）：

- 宏平均 F1 值（macro-averaging F1 score）：对每个类别单独计算 F1 值，最后取平均得到 F_{macro}
- 微平均 F1 值（micro-averaging F1 score）：用每个类别上的全部 TP, FP, FN 样本直接计算微精确率和微召回率，最后用上面公式计算得到 F_{micro}

对于 N 类检测问题（$N > 2$）：

- 宏平均 F1 值（macro-averaging F1 score）

$$F_{\text{macro}} = \frac{1}{N} \sum_{n=1}^{N} F_{1,n}$$

- 微平均 F1 值（micro-averaging F1 score）

$$P_{\text{micro}} = \frac{\sum_{n=1}^{N} \text{TP}_n}{\sum_{n=1}^{N} \text{TP}_n + \sum_{n=1}^{N} \text{FP}_n}$$

$$R_{\text{micro}} = \frac{\sum_{n=1}^{N} \text{TP}_n}{\sum_{n=1}^{N} \text{TP}_n + \sum_{n=1}^{N} \text{FN}_n}$$

$$F_{\text{micro}} = \frac{2 \times P_{\text{micro}} \times R_{\text{micro}}}{P_{\text{micro}} + R_{\text{micro}}}$$

Q: macro F1 可能存在什么问题？

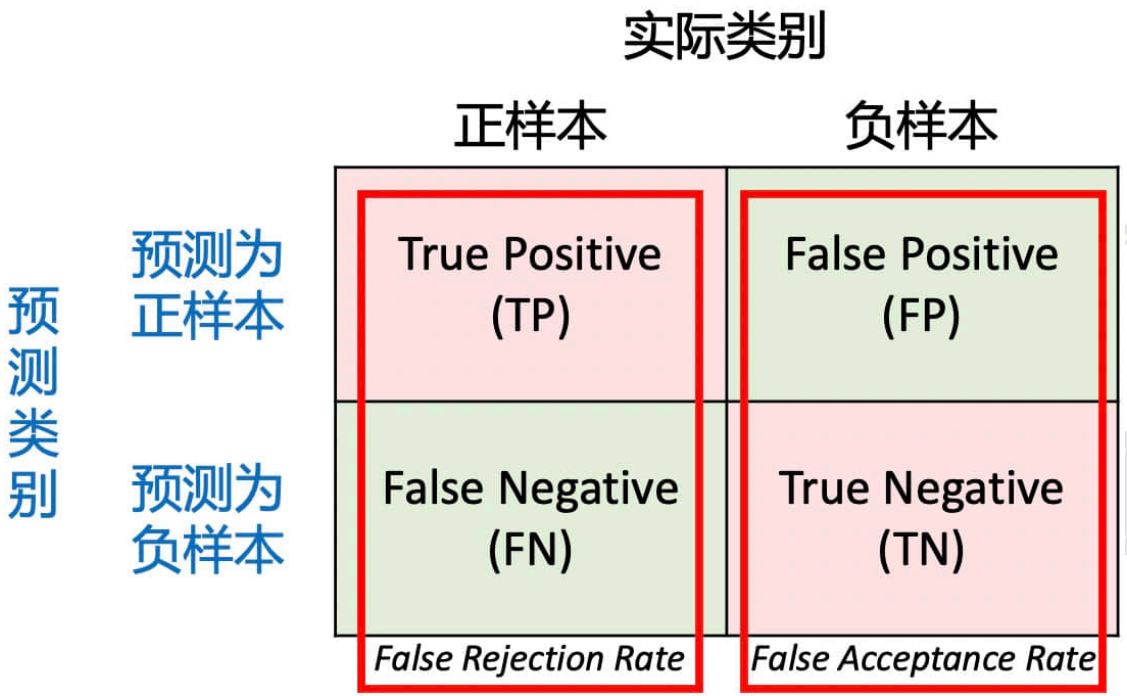
分类/检测问题中的评估指标

误识率（FAR）和误拒率（FRR）

误识率（False Acceptance Rate, FAR）/FPR (False Positive Rate)：在所有负样本中，被预测为正样本的比例

误拒率（False Rejection Rate, FRR）/FNR（False Negative Rate)：在所有正样本中，被预测为负样本的比例

$$\mathrm{FAR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}, \quad \mathrm{FRR} = \frac{\mathrm{FN}}{\mathrm{FN} + \mathrm{TP}}$$



分类/检测问题中的评估指标

误识率（FAR）和误拒率（FRR）

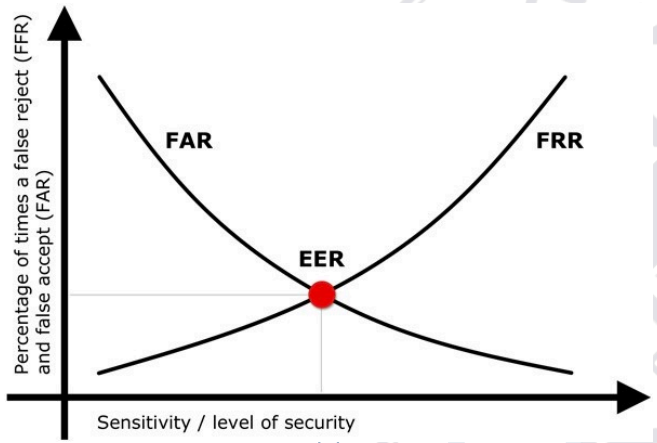
误识率（False Acceptance Rate, FAR）/FPR (False Positive Rate)：在所有负样本中，被预测为正样本的比例

$$\mathrm{FAR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}$$

误拒率（False Rejection Rate, FRR）/FNR（False Negative Rate)：在所有正样本中，被预测为负样本的比例

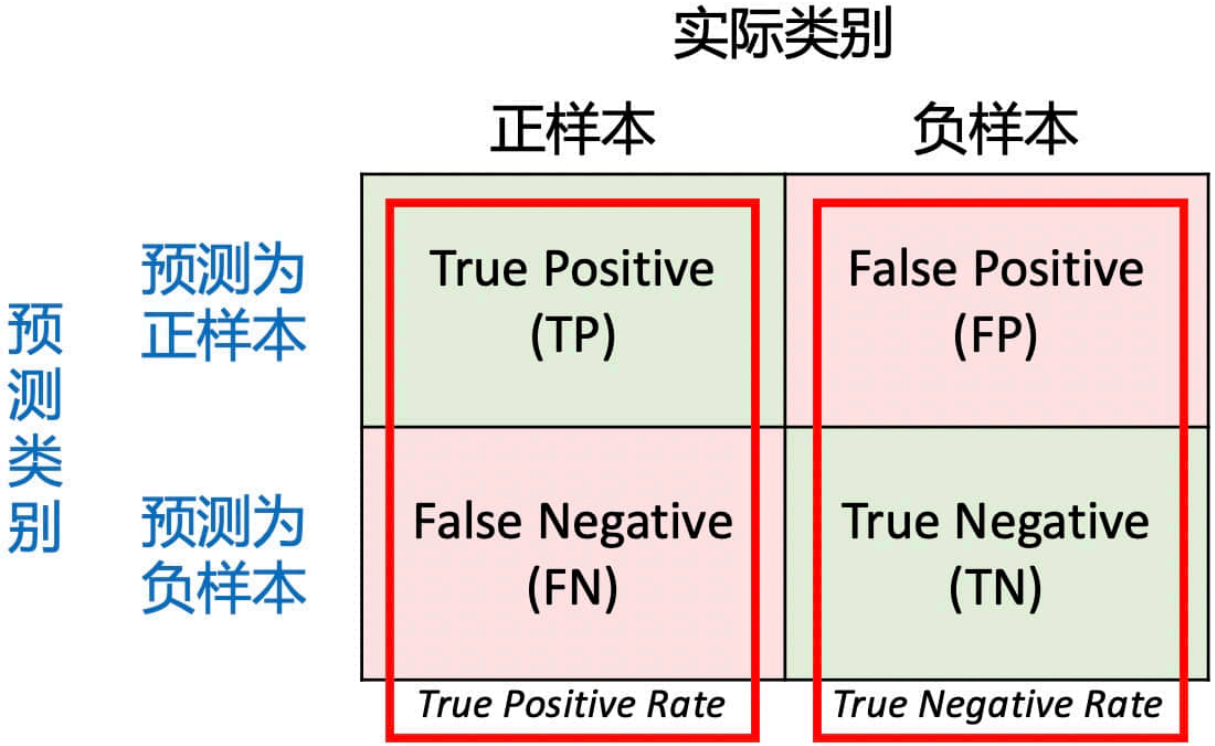
$$\mathrm{FRR} = \frac{\mathrm{FN}}{\mathrm{FN} + \mathrm{TP}}$$

等错误率（Equal Error Rate, EER)：通过调整阈值使得 FAR 与 FRR 相等时的取值



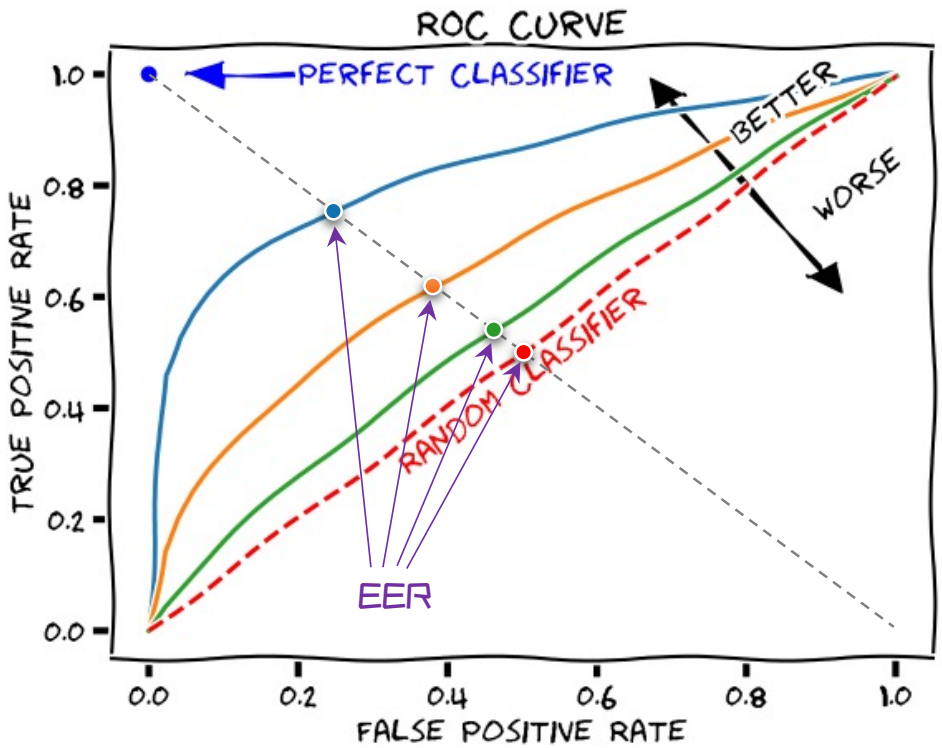
ROC 曲线 (Receiver Operating Characteristic curve)：在不同阈值下，TPR (True Positive Rate) 和 FPR (False Positive Rate) 组成的曲线

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \qquad \mathrm{TNR} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}, \quad \mathrm{FPR} = 1 - \mathrm{TNR}$$



ROC 曲线 (Receiver Operating Characteristic curve)

- 等错误率（Equal Error Rate, EER）：ROC 曲线与 y=1-x 交点的横坐标 (FPR)



AUC (Area Under Curve)：ROC 曲线下的面积

EVALUATING VAD FOR AUTOMATIC SPEECH RECOGNITION

- 指出传统评估指标的不足：大部分指标基于帧级别语音/非语音的分类准确率，可能导致 VAD 与下游 ASR 任务性能之间的相关性很弱，不能很好地评估 VAD 对于 ASR 的影响
- 除帧级别准确率之外，还考虑多种边界效应，提出了一种新的评估指标 VACC (VAD Accuracy)：

$$J_{\text{VAD}} = \frac{4}{\frac{1}{\mathcal{J}_A} + \frac{1}{\mathcal{J}_S} + \frac{1}{\mathcal{J}_E} + \frac{1}{\mathcal{J}_B}}$$

  - 帧级别准确率：$\mathcal{J}_A$
  - 左边界准确率（Start boundary accuracy, SBA）：$\mathcal{J}_S$
  - 右边界准确率（End boundary accuracy, EBA）：$\mathcal{J}_E$
  - 边界精确率（Border Precision, BP）：$\mathcal{J}_B$

Recap: 统计语音识别

大词汇连续语音识别系统架构

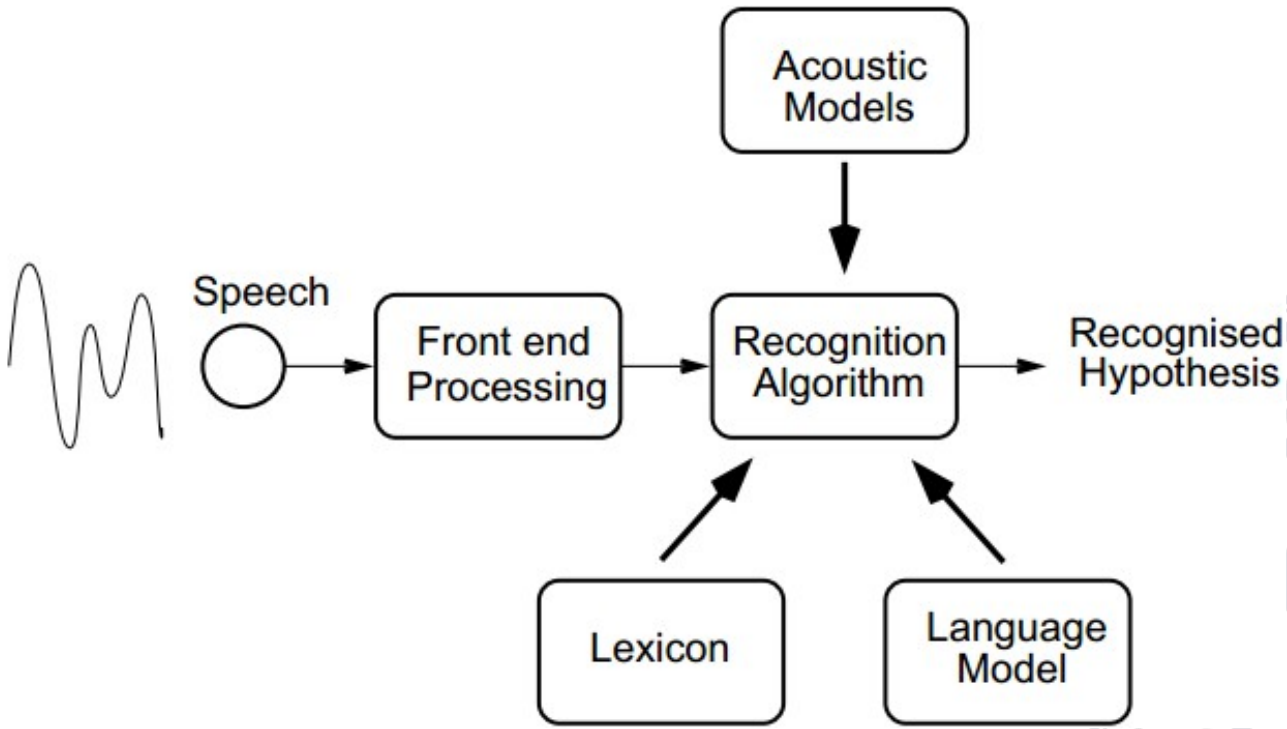
$$\hat{\mathbf{W}} = \arg\max_{\mathbf{W}} p(\mathbf{A}|\mathbf{O})p(\mathbf{O}|\mathbf{L})P(\mathbf{L}|\mathbf{W})P(\mathbf{W})$$

W: 词序列

L: 声学建模单元序列

O: 音频样本序列

A: 声学特征序列



声学模型 $p(\mathbf{O}|\mathbf{L})$

声学模型是一个**概率**模型，它可以刻画用来描述不同声音的声学特性。

- 语音识别最关键的技术之一
- 概率模型 $p(\mathbf{O}|\mathbf{L})$ 用于刻画不同语音单元，如音素，音节，字，词
- **Hidden Markov Model (HMM)** 隐马尔科夫模型由于其概念的简单以及数学的完备，被最广泛地采用。

HMM 可以认为是一个最基本的有限状态转录机（FST），它可以将一个用于表示语音的特征向量序列，通过有限状态机，转换成状态机的状态序列（表示音素、音节或词）。

语言模型 $P(\mathbf{W})$

语言模型是一个**概率模型 probabilistic** model：

1. 引导搜索算法 (在给定历史的情况下预测下一个词)。
2. 消除声学单元之间的混淆性，特别是那些声学层相似的单元。

   `Great wine v.s. Grey twine`

语言模型将概率分配到一串要识别的 tokens（通常是词）上：

- **上下文自由语法**:

  `( <s> < one | two | three > </s> )`

- **统计语言模型**: n-gram 语言模型

$$P(w_1, w_2, \cdots, w_N)$$

n-gram 统计语言模型和 HMM 声学模型由于容易结合而被广泛地用于语音识别中。

字典模型 $P(\mathbf{L}|\mathbf{W})$

字典模型为声学模型和语言模型之间构建了桥梁。

- 它在词和声学单元之间定义了一个映射。
- 它可以是一个**确定化**的模型 (**deterministic**)

Word	Pronunciation
TOMATO	t ah m aa t ow t ah m ey t ow
COVERAGE	k ah v er ah jh k ah v r ah jh

- 它也可以是一个**概率**模型 (**probabilistic**)

Word	Pronunciation	Probability
TOMATO	t ah m aa t ow t ah m ey t ow	0.45 0.55
COVERAGE	k ah v er ah jh k ah v r ah jh	0.65 0.35

统计语音识别

总结

- **特征提取 Feature extraction** $p(\mathbf{A}|\mathbf{O})$ **(前端处理)**
- **声学模型 Acoustic model** $p(\mathbf{O}|\mathbf{L})$**:** 在给定声学单元（如音素）的条件下对特征分布进行建模。
- **字典 Lexicon** $P(\mathbf{L}|\mathbf{W})$**:** 声学建模单元和词之间的映射。
- **语言模型 Language model** $P(\mathbf{W})$**:** 产生词序列的概率。

$$\hat{\mathbf{W}} = \arg\max_{\mathbf{W}} p(\mathbf{A}|\mathbf{O})p(\mathbf{O}|\mathbf{L})P(\mathbf{L}|\mathbf{W})P(\mathbf{W})$$

- **解码算法 Decoding algorithm:** 基于以上各种信息源，找到“最优”词序列。
- **结果评估 Evaluation:** 对比 *hypotheses*（识别假设标注）和 *reference*（参考标注）

ASR 系统性能评估

ASR 系统性能评估：通过比较第 i^{th} 句话的识别假设标注 H_i 和它的真实参考标注 R_i，来统计得到最后的识别性能。

- 句子错误率 Sentence error rate (SER)
- 词错误率 Word error rate (WER) / 字错误率 Character error rate (CER)
- 音素错误率 Phone error rate (PER) / 音节错误率 Syllable error rate (YER)

SER 定义:

$$\text{SER} = \frac{1}{N}\sum_{i=1}^{N}\delta(H_i, R_i) \times 100\% \quad \delta(H_i, R_i) = \begin{cases} 1 & H_i = R_i \\ 0 & H_i \neq R_i \end{cases}$$

其中 N 是测试集合的总语句数。

Levenshtein Distance

- 计算两个序列之间的差距 (a.k.a. 编辑距离)
- 定义：将一个序列变成另一个序列所需要的最少修改次数
- **动态规划**算法可以用于计算编辑距离

WER/CER/PER/YER 计算如下：

$$\text{WER} = \frac{\sum_{i=1}^{N}(I_i + D_i + S_i)}{\sum_{i=1}^{N} T_i}$$

其中 I_i, D_i 和 S_i 分别是通过编辑距离算法计算每一对 (R_i, H_i) 后得到的各种错误的数量，包括：插入 insertions, 删除 deletions 和替代 substitutions。T_i 是真实参考标注 R_i 中的单元总数（如: 词，音素，词，音节）。

Q: WER 能否大于 100%？

Levenshtein Distance 算法

```
int LevenshteinDistance(char s[], char t[])
    int m = s.length, n = t.length;
    int d[][] = new int[m+1][n+1];
    for (int i=0; i<=m; i++) d[i][0]= i;
    for (int j=0; j<=n; j++) d[0][j]= j;
    for (int i=0; i<m; i++) {
        for (int j=0; j<n; j++) {
            int cost = (s[i] == t[j]) ? 0 : 1;
            d[i+1][j+1] = minimum (d[i-1][j]   + 1,        // deletion
                                   d[i][j-1]   + 1,        // insertion
                                   d[i-1][j-1] + cost);    // substitution
        }
    }
    return d[m][n];
}
```

Levenshtein Distance 示例

SATURDAY v.s. SUNDAY

		S	A	T	U	R	D	A	Y
	0	1	2	3	4	5	6	7	8
S	1								
U	2								
N	3								
D	4								
A	5								
Y	6								

- 初始化：填满第一行和第一列
  - 第一行：插入（insertion）错误的数量
  - 第一列：删除（deletion）错误的数量
- 参考标注：SUNDAY
- 假设标注：SATURDAY

Levenshtein Distance 示例

SATURDAY v.s. SUNDAY

		S	A	T	U	R	D	A	Y
	0	1	2	3	4	5	6	7	8
S	1	0	1						
U	2								
N	3								
D	4								
A	5								
Y	6								

- 考虑 entry $(i = 1, j = 1)$
  - **插入:** d[1][0]+1=2
  - **删除:** d[0][1]+1=2
  - **替代:** d[0][0]+0=0
- 最小代价: cost=0

- 考虑 entry $(i = 1, j = 2)$
  - **插入:** d[1][1]+1=1
  - **删除:** d[0][2]+1=2
  - **替代:** d[0][1]+1=2
- 最小代价: cost=1

Levenshtein Distance 示例

SATURDAY v.s. SUNDAY

		S	A	T	U	R	D	A	Y
	0	1	2	3	4	5	6	7	8
S	1	**0**	**1**	**2**	3	4	5	6	7
U	2	1	1	2	**2**	3	4	5	6
N	3	2	2	2	3	**3**	4	5	6
D	4	3	3	3	3	4	**3**	4	5
A	5	4	3	4	4	4	4	**3**	4
Y	6	5	4	4	5	5	5	4	**3**

因此, 最优的 *Levenshtein distance* 是 3 (2 insertions and 1 substitution)

REF	S			U	N	D	A	Y
HYP	S	A	T	U	R	D	A	Y
Edits		I	I		S			

语音识别的相关开源工具

- **Hidden Markov Toolkit:**
  http://htk.eng.cam.ac.uk

- **Kaldi:** Acoustic model, decoder + language model and DNN (recently)
  http://kaldi.sourceforge.net

  参考资料:
  - https://kaldi-asr.org/doc/
  - 《Kaldi 语音识别实战》 电子工业出版社 陈果果等
  - 《语音识别基本法—Kaldi 实践与探索》 清华大学语音和语言技术中心 汤志远等
    (http://cslt.riit.tsinghua.edu.cn/mediawiki/images/2/25/Speech_book.pdf)

Kaldi 相关介绍

Kaldi — Speech recognition toolkit[1]

- **功能:**
  - 各种常规的基于 HMM 的声学建模方法 (GMM/CNN/LSTM/TDNN-HMM)
  - 基于 nnet1/nnet2/nnet3 的鉴别性训练
  - 丰富且具有完整流程的数据集示例
  - 还支持其他语音相关任务，如关键词搜索与语音唤醒、语音端点检测、说话人识别等
- **技术特性:**
  - 源代码库由 C++ 代码写成，平台包括 Linux, Mac, Windows
  - 代码容易阅读和理解，易于修改和拓展
  - 大量的线性代数的支持，易于在不同线性代数库之间切换
  - 代码级集成有限状态转录机（FST）技术，具有基于 FST 的现代解码器

[1] https://github.com/kaldi-asr/kaldi

Kaldi: 编译过程 (Linux)

- 下载最新版本源代码（5.5）

```
git clone https://github.com/kaldi-asr/kaldi.git
```

- 检查编译依赖库（按照脚本输出提示安装缺失的库）

```
cd kaldi
tools/extras/check_dependencies.sh
```

- 安装第三方工具

```
cd tools
make openfst  # install openfst
make cub      # install CUDA programming support
...
```

Kaldi: 编译过程-续 (Linux)

- 编译 Kaldi 代码
  (参考 https://github.com/kaldi-asr/kaldi/blob/master/INSTALL)

```
cd src

# run "./configure --help" for more information
./configure --shared --use-cuda=no  # only use CPU

# compile with 8 threads
make depend -j 8
make -j 8
```

- 编译完成后，生成的可执行文件都存放在各自的代码目录下，如 bin、featbin 等，可以在环境变量 PATH 中增加这些目录的路径以方便地调用 Kaldi 的工具。

要获取每个 Kaldi 命令工具的使用信息，直接在命令行执行该工具（不加任何参数）即可。如 `paste-feats`

```
$ paste-feats

Paste feature files (assuming they have about the same durations,
see --length-tolerance), appending the features on each frame;
think of the unix command 'paste'.
Usage: paste-feats <in-rspecifier1> <in-rspecifier2> [<in-rspecifier3> ...] <out-wspecifier>
 or: paste-feats <in-rxfilename1> <in-rxfilename2> [<in-rxfilename3> ...] <out-wxfilename>
 e.g. paste-feats ark:feats1.ark "ark:select-feats 0-3 ark:feats2.ark ark:- |" ark:feats-out.ark
  or: paste-feats foo.mat bar.mat baz.mat
See also: copy-feats, copy-matrix, append-vector-to-feats, concat-feats

Options:
  --binary                    : If true, output files in binary (only relevant for single-file operation,
  --length-tolerance          : If length is different, trim as shortest up to a frame  difference of leng

Standard options:
  --config                    : Configuration file to read (this option may be repeated) (string, default
  --help                      : Print out usage message (bool, default = false)
  --print-args                : Print the command line arguments (to stderr) (bool, default = true)
  --verbose                   : Verbose level (higher->more logging) (int, default = 0)
```

Kaldi: 常用命令/脚本

特征提取

kaldi/src/featbin/目录下的命令：

- 计算语谱图：compute-spectrogram-feats
- 计算 FBank 特征：compute-fbank-feats
- 计算 MFCC 特征：compute-mfcc-feats
- etc.

提取某一条音频的 FBank 特征，并以文本格式 (t) 输出到标准输出 (-)：

```
compute-fbank-feats --config=conf/fbank.conf \
  'scp:echo "utt1 wavdata/128-134883-0045.wav" |' ark,t:-
```

提取某一条音频的 FBank 特征，并以文本格式 (t) 输出到标准输出 (-)：

```
$ compute-fbank-feats --config=conf/fbank.conf 'scp:echo "utt1 wavdata/128-134883-0045.wav" |' ark,t:-

utt1  [
  -1.785322 -0.5190077 -0.3890214 -0.2648814 0.2216281 ... 8.571596 8.001829

  -0.06027339 0.6006252 0.5203104 0.665014 0.7794935 ... 8.761829 8.26743

  ...

  5.729714 5.614149 7.239074 8.069734 8.08042 7.707728 ... 11.97327 11.54237

  13.45795 13.04935 12.27463 12.04241 11.6731 12.63377 ... 9.827711 9.869823 ]
```

配置文件 conf/fbank.conf 内容如下：

```
--sample-frequency=16000
--num-mel-bins=80
```

Kaldi: Transcription and Hypotheses

- **Transcription**: 训练时已知的文本信息，通常基于词，也有基于字符或者音素等的。
- **Hypothesis:** ASR 系统的文本输出。类似地，它可以是基于词，字符或音素的。

Kaldi 采用文本表单格式来表示标注文本：

Trans: ref.txt

```
440c0407 MANY WANT TO STORM THE TANK AND TAKE IT OVER
440c0409 GRAINS AND SOYBEANS MOST CORN AND WHEAT FUTURES PRICES WERE STRONGER
447c040z YIELD MANAGEMENT ISN'T ALL BAD FOR CONSUMERS
447c0412 THE FEE ON THE GREEN OR BASIC CARD WILL JUMP TO FIFTY FIVE DOLLARS
```

Hyp: hyp.txt

```
440c0407 MANY WANT TO STORM THE TANK AND TAKEN OVER
440c0409 GRAINS AND SOYBEANS MOST CORN AND WHEAT FUTURES PRICES WERE STRONGER
447c040z YIELD MANAGEMENT ISN'T ALL BAD FOR CONSUMERS
447c0412 THE FEE ON THE GRAY AND OUR BASIC CARD WILL JUMP TO FIFTY FIVE DOLLARS
```

通过比较文本转写标签和识别假设标注来得到词错误率（WER）：

```
compute-wer --text --mode=present \
    ark,t:/path/to/ref.txt ark,t:/path/to/hyp.txt
```

```
$ compute-wer --text --mode=present ark,t:/path/to/ref.txt ark,t:/path/to/hyp.txt

%WER 11.90 [ 5 / 42, 1 ins, 1 del, 3 sub ]
%SER 50.00 [ 2 / 4 ]
Scored 4 sentences, 0 not present in hyp.
```

- **SER: 一定小于等于 100%**

$$\text{SER} = \frac{H}{N} \times 100\%$$

- **WER: 可能大于 100%**

$$\text{WER} = \frac{S + D + I}{N} \times 100\%$$

评估系统性能

输出单词对齐

```
align-text "--special-symbol='***'" \
   ark:/path/to/ref.txt ark:/path/to/hyp.txt ark,t:- | \
   utils/scoring/wer_per_utt_details.pl --special-symbol "'***'"
```

```
$ align-text --special-symbol="'***'" \
   ark:/path/to/ref.txt ark:/path/to/hyp.txt ark,t:- | \
   utils/scoring/wer_per_utt_details.pl --special-symbol "'***'"

LOG (align-text[5.4.215-a630d]:main():align-text.cc:129) Done 4 sentences, failed for 0
utils/scoring/wer_per_utt_details.pl: Note: handling as utf-8 text
440c0407 ref  MANY  WANT  TO  STORM  THE  TANK  AND   TAKE    IT   OVER
440c0407 hyp  MANY  WANT  TO  STORM  THE  TANK  AND  '***'  TAKEN  OVER
440c0407 op     C     C    C    C     C     C    C     D      S      C
440c0407 #csid 8 1 0 1
440c0409 ref  GRAINS  AND  SOYBEANS  MOST  CORN  AND  WHEAT  FUTURES  PRICES  WERE  STRONGER
440c0409 hyp  GRAINS  AND  SOYBEANS  MOST  CORN  AND  WHEAT  FUTURES  PRICES  WERE  STRONGER
440c0409 op      C     C       C       C     C    C     C       C        C      C       C
440c0409 #csid 11 0 0 0
447c040z ref  YIELD  MANAGEMENT  ISN'T  ALL  BAD  FOR  CONSUMERS
447c040z hyp  YIELD  MANAGEMENT  ISN'T  ALL  BAD  FOR  CONSUMERS
447c040z op     C         C        C     C    C    C       C
447c040z #csid 7 0 0 0
447c0412 ref  THE  FEE  ON  THE  '***'  GREEN   OR  BASIC  CARD  WILL  JUMP  TO  FIFTY  FIVE  DOLLARS
447c0412 hyp  THE  FEE  ON  THE   GRAY   AND   OUR  BASIC  CARD  WILL  JUMP  TO  FIFTY  FIVE  DOLLARS
447c0412 op    C    C    C   C     I      S     S     C      C     C     C    C     C      C      C
447c0412 #csid 12 2 1 0
```