

语音识别：从入门到精通

第八讲：基于WFST的解码器



主讲人 吕航

西北工业大学
博士

hanglyu1991@gmail.com





内容提要

- 解码
 - 公式
 - 框架
 - 孤立词与LM
 - 计算问题
- 解码器
 - 剪枝(Pruning)
 - Lattice和N-best List
 - A* decoder思想(简要)
 - 字典树(简要)
 - Language Look-ahead(简要)
- WFST介绍
- 识别中的WFST—基于kaldi



解码任务—解码公式

给定声学观测 $O = o_1, o_2, \dots, o_T$, 找到最可能的词序列 $W = w_1, w_2, \dots, w_N$:

$$\begin{aligned}\hat{W} &= \operatorname{argmax}_w P(W|O) \\ &= \operatorname{argmax}_w \underbrace{P(O|W)}_{\text{Acoustic Model}} \underbrace{P(W)}_{\text{Language Model}}\end{aligned}$$

$$\hat{W} = \operatorname{argmax}_w P(O|W)P(W)$$

实际应用中 对么？

其实你已经学过理论上该怎么做了！----Viterbi算法。

(顺便回忆一下Forward algorithm和Viterbi algorithm的关系)

找到了最可能的状态序列，我们就能恢复出最可能的词序列。

---解码范围，LM约束的图



解码任务—解码公式

给定声学观测 $O = o_1, o_2, \dots, o_T$, 找到最可能的词序列 $W = w_1, w_2, \dots, w_N$:

$$\begin{aligned}\hat{W} &= \operatorname{argmax}_w P(W|O) \\ &= \operatorname{argmax}_w \underbrace{P(O|W)}_{\text{Acoustic Model}} \underbrace{P(W)}_{\text{Language Model}}\end{aligned}$$



$$= \operatorname{argmax}_w P(O|W) P(W)^{LMWT}$$



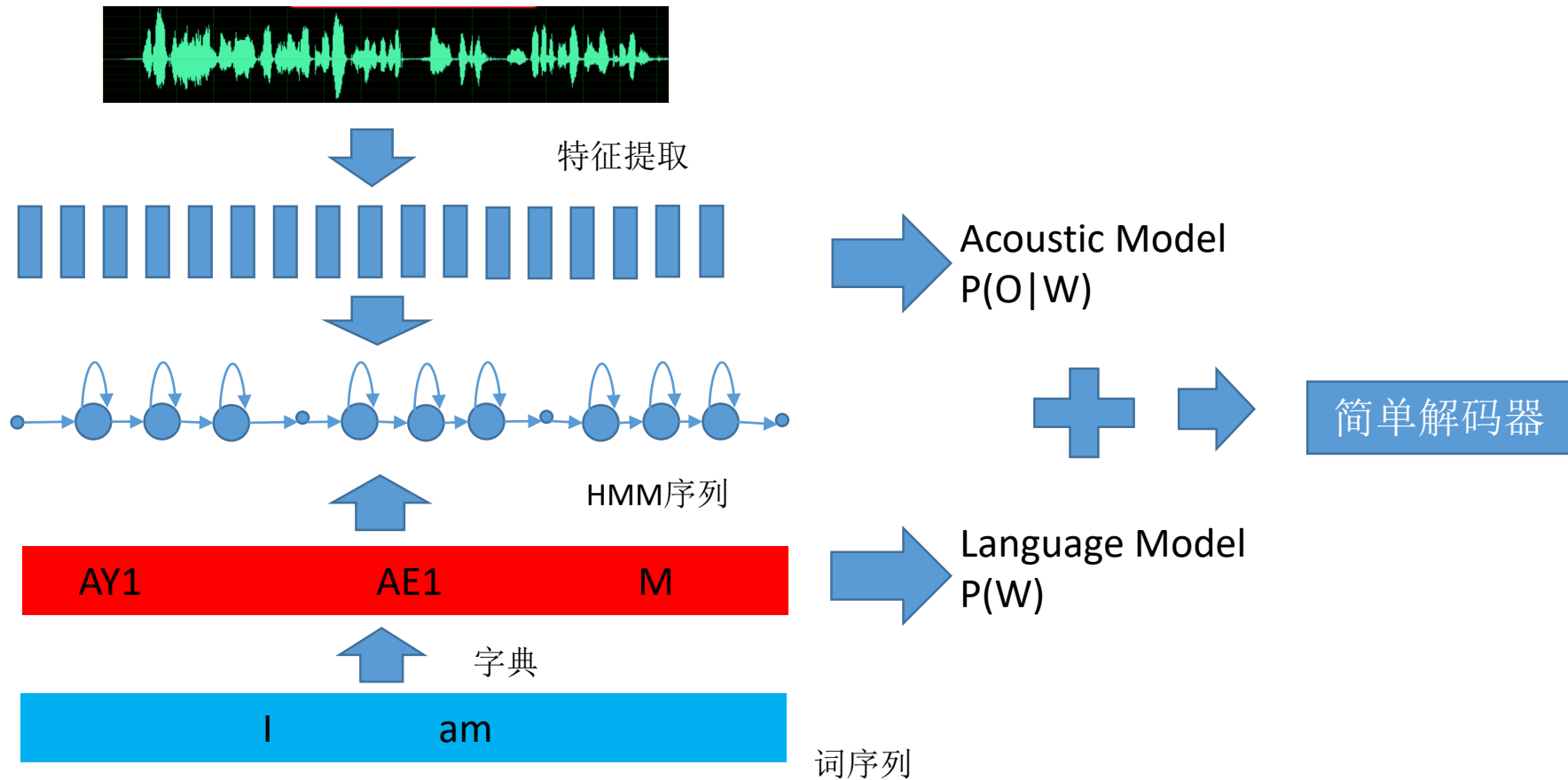
语言模型缩放权重(LMWT): language model weight

$$= \operatorname{argmax}_w P(O|W) P(W)^{LMWT} WIP^N$$

插入词惩罚(WIP): word insertion penalty



解码任务一框架





解码任务—Token Passing Algorithm(令牌环传递算法)

Token Passing算法就是Viterbi算法的实现

Token的设计：存储经过某状态的最优路径的概率，存储与全局最优路径的距离，以及帮助寻找其它token或者回溯的指针。

Initialisation:

Each model initial state holds a token with value 0;

All other states hold a token with value ∞

Algorithm:

for $t := 1$ to T **do**

for each state i **do**

 Pass a copy of the token in state i to all connecting
 states j , incrementing its s value by $p_{ij} + d_j(t)$;

end;

 Discard the original tokens;

for each state i **do**

 find the token in state i with the smallest s
 value and discard the rest

end;

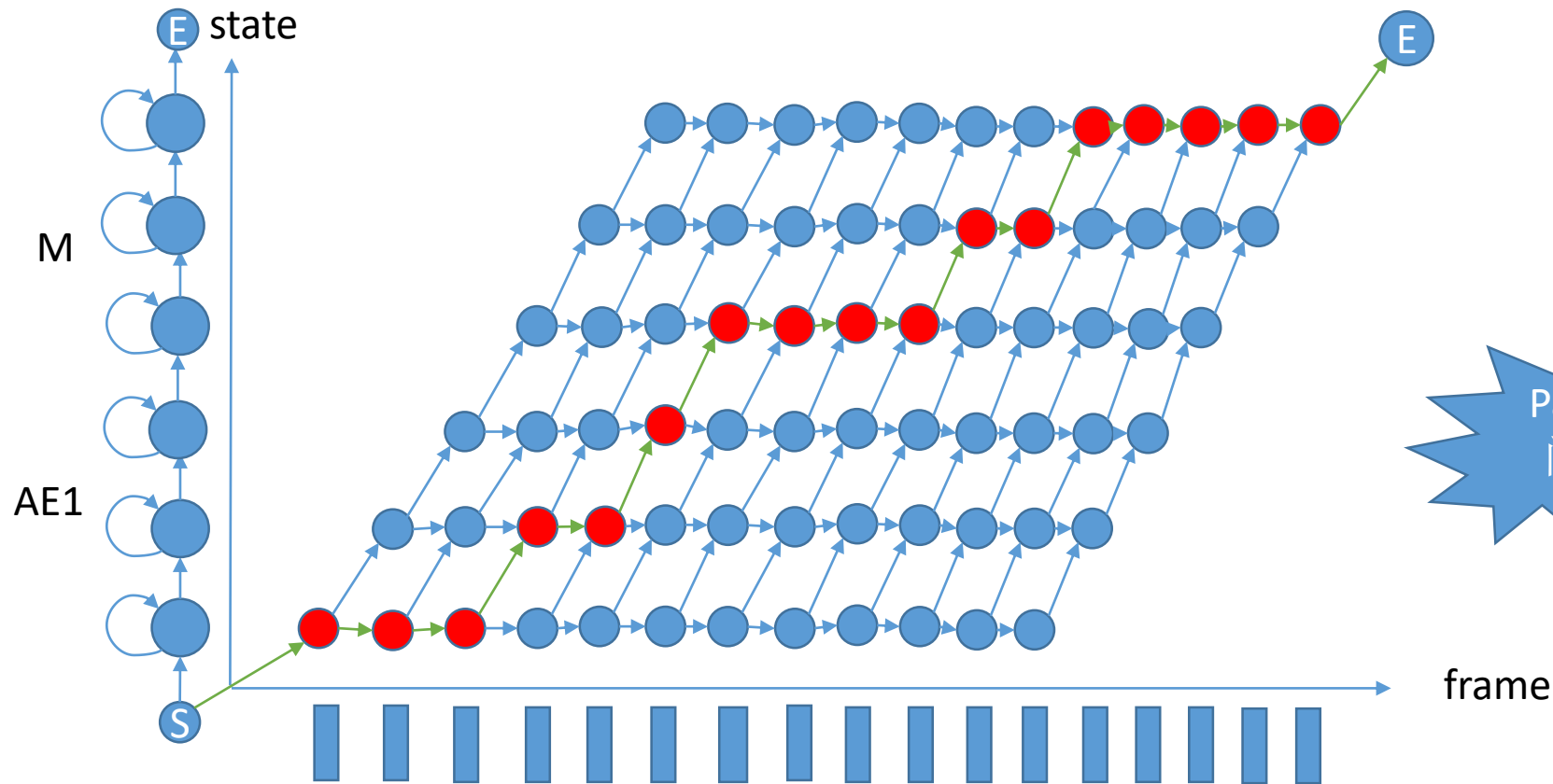
end;

发射概率

转移概率

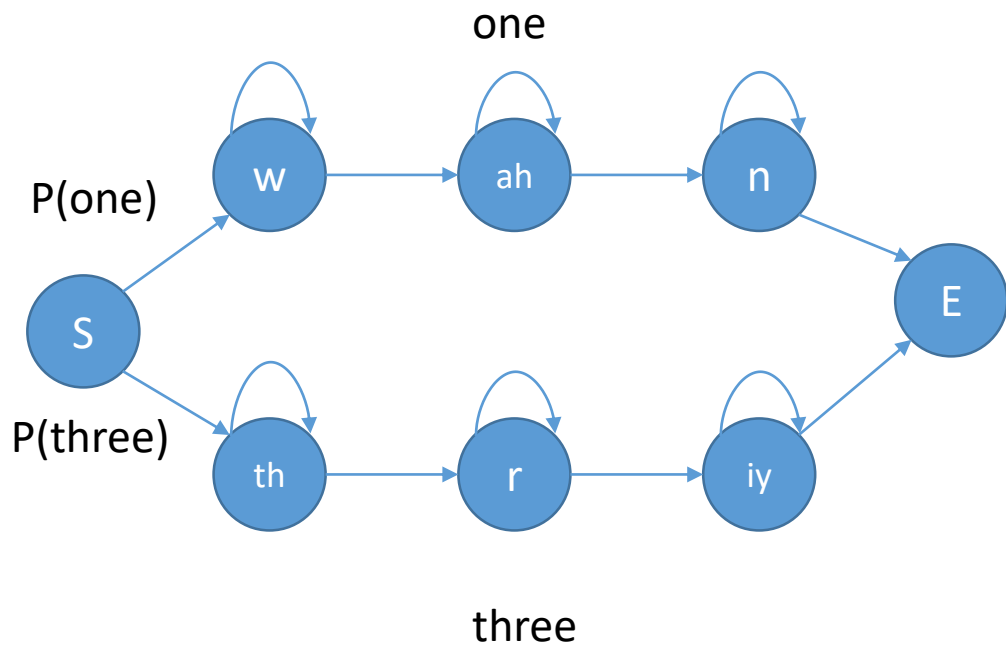


解码任务—晶格网 (trellis)





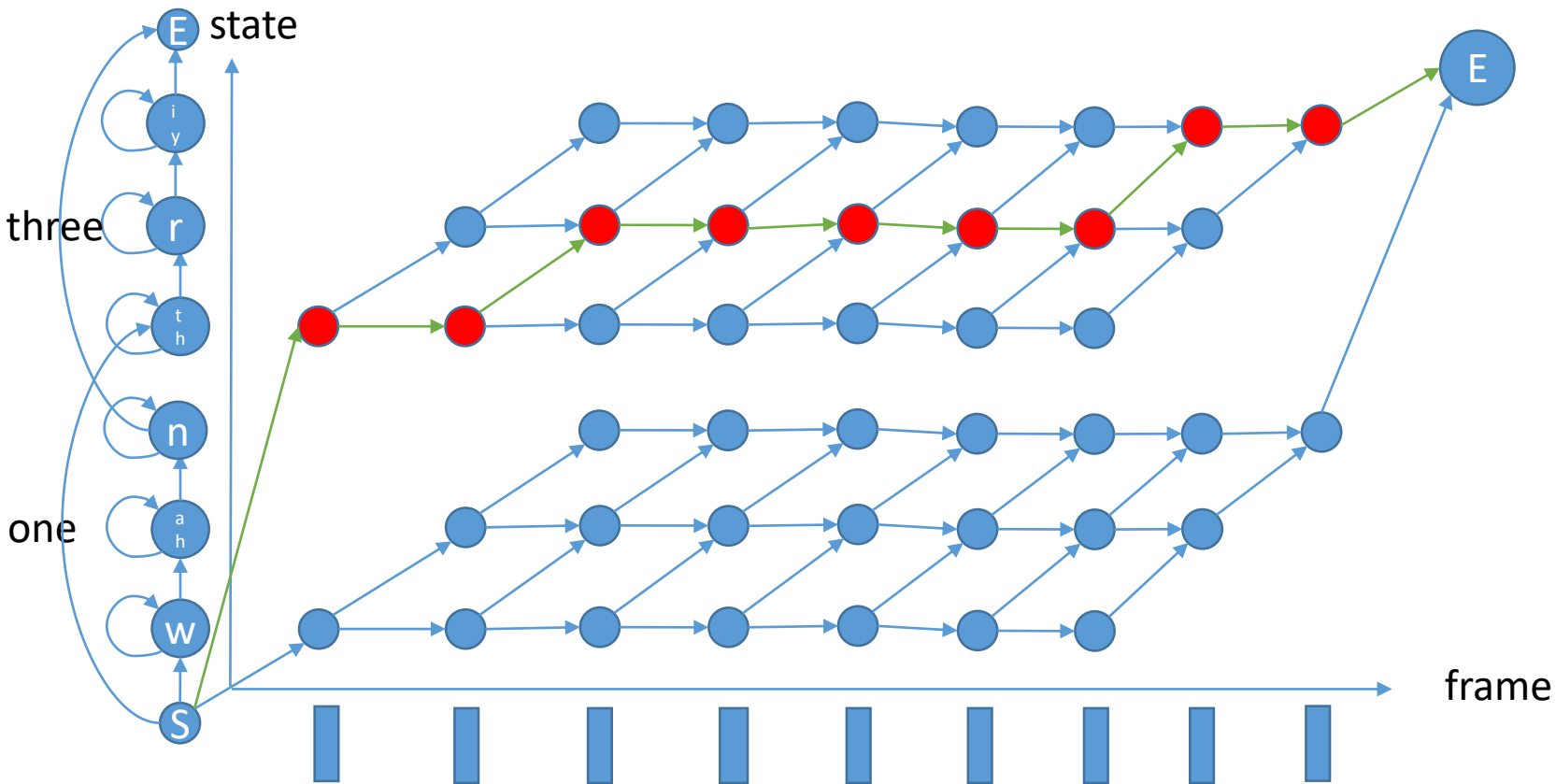
解码任务—孤立词



示意图：每个phone一个状态(for simple)



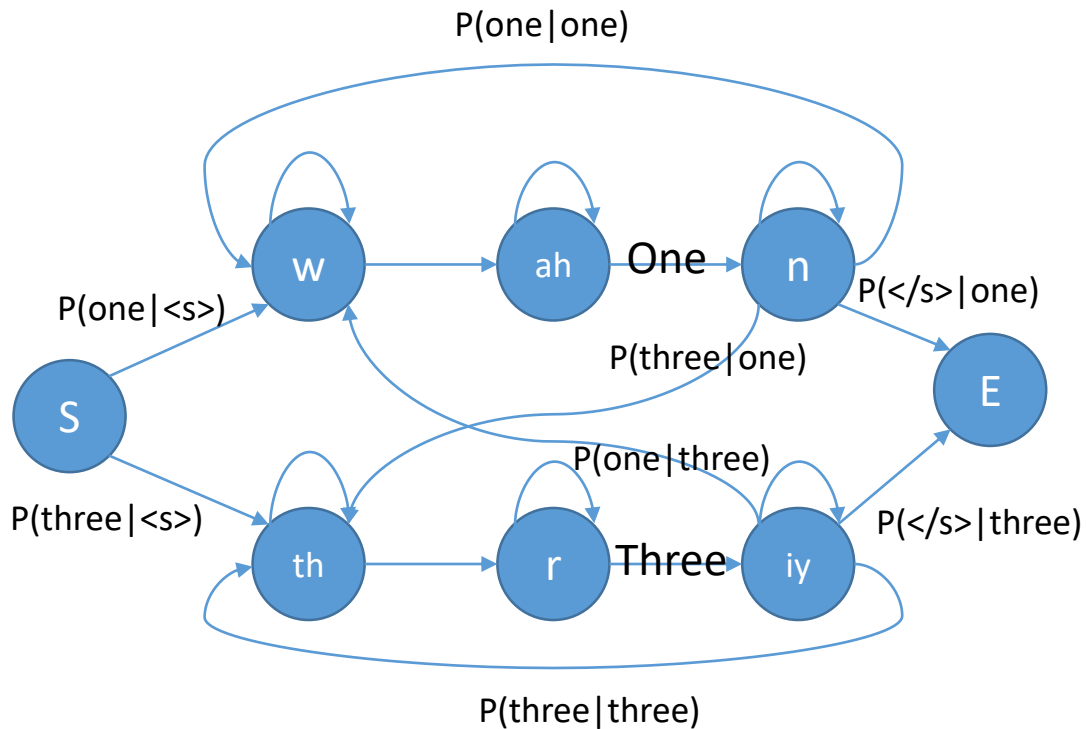
解码任务—孤立词





解码任务—LM

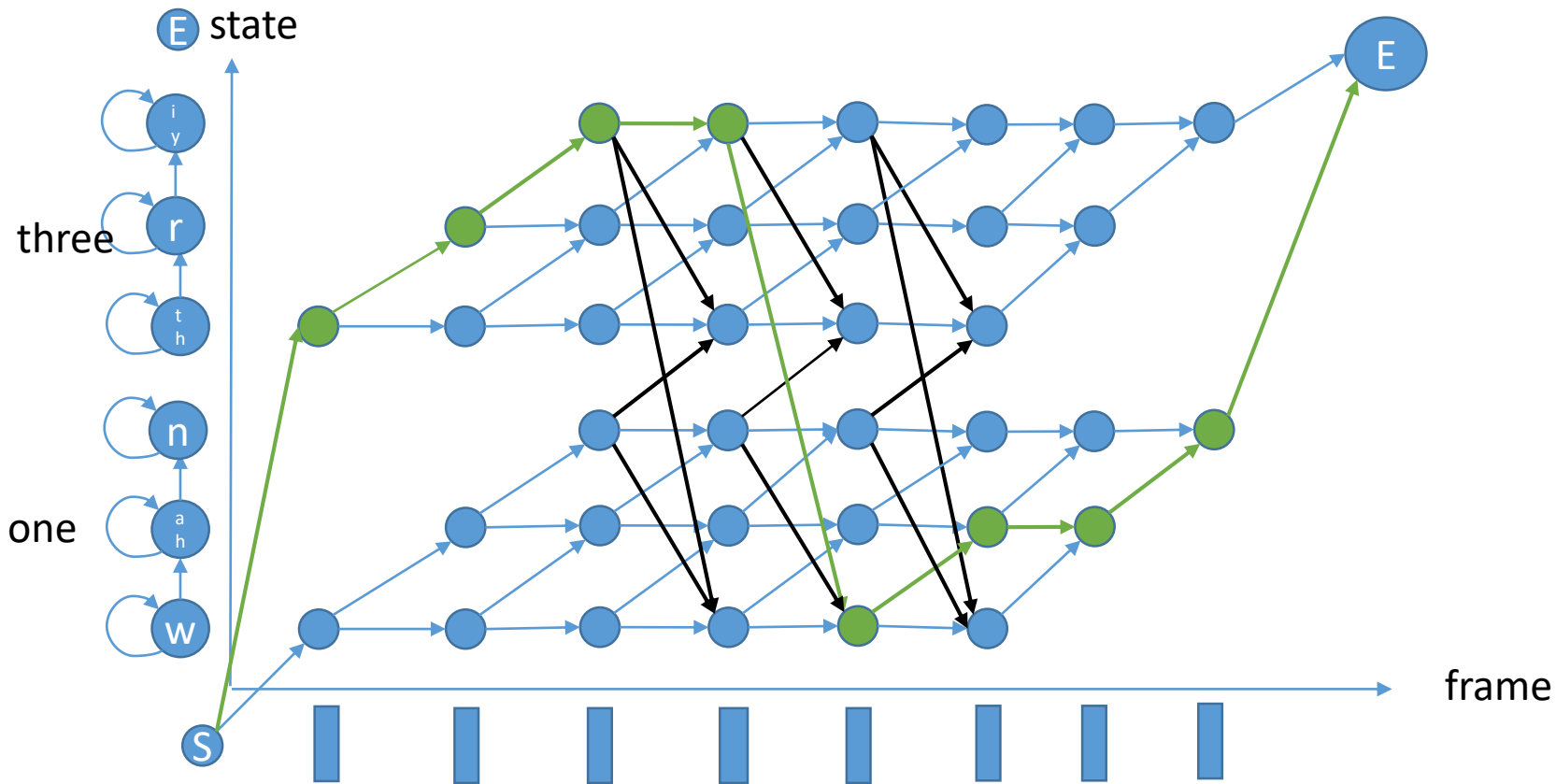
Bigram为例子



示意图：每个phone一个状态(for simple)



解码任务—LM





Viterbi算法理论上准确有效的完成了解码工作，实际上有什么问题？

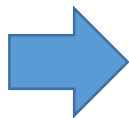
你已经训练过LM了，它的wordlist是不是巨大？形成的N-gram是不是条数超多？

由于LVCSR实际任务中N-gram，Lexicon和triphone建模导致无法只简单的使用Viterbi算法，需要进行一些工程优化。



如何解决计算问题？

1. 剪枝(Pruning):beam search and histogram pruning
2. 多阶段解码(Multi-stage decoding)
3. A*解码(A* decoding)
4. 树状字典(Tree structured lexicons)
5. 语言模型超前使用(Language model Look-ahead)
-



解码器



思想：去除没有竞争力的路径。

Beam Search: 每帧只保留Best Path以及与Best Path距离小于beam-threshold的tokens。

Histogram Pruning: 每帧只处理前Top N个tokens。



产生一条最优路径只是解码器的部分工作，对于解码器的研究，更重要的是生成一个准确的lattice，然后再进行后处理，如重打分(Re-scoring)[i.e. 多阶段解码]。

N-best List: 解码获得最好的Top N条词序列。

Lattice: 他是一个有向图，有效的表示关于可能词序列的更多信息。[i.e. 把到达终点的tokens走过路径的信息绘制在一张图里。]

Rank	Path	AM logprob	LM logprob
1.	it's an area that's naturally sort of mysterious	-7193.53	-20.25
2.	that's an area that's naturally sort of mysterious	-7192.28	-21.11
3.	it's an area that's not really sort of mysterious	-7221.68	-18.91
4.	that scenario that's naturally sort of mysterious	-7189.19	-22.08
5.	there's an area that's naturally sort of mysterious	-7198.35	-21.34
6.	that's an area that's not really sort of mysterious	-7220.44	-19.77
7.	the scenario that's naturally sort of mysterious	-7205.42	-21.50
8.	so it's an area that's naturally sort of mysterious	-7195.92	-21.71
9.	that scenario that's not really sort of mysterious	-7217.34	-20.70
10.	there's an area that's not really sort of mysterious	-7226.51	-20.01

Figure 10.2 An example 10-Best list from the Broadcast News corpus, produced by the CU-HTK BN system (thanks to Phil Woodland). Logprobs use \log_{10} ; the language model scale factor (LMSF) is 15.

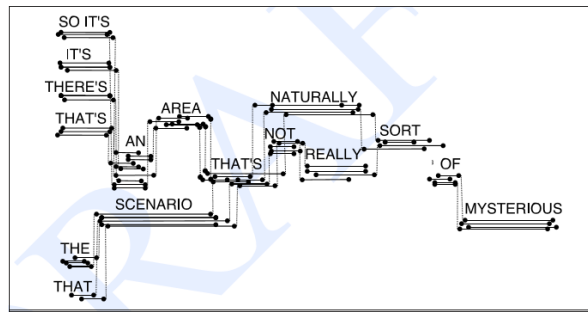


Figure 10.3 Word lattice corresponding to the N-best list in Fig. 10.2. The arcs beneath each word show the different start and end times for each word hypothesis in the lattice; for some of these we've shown schematically how each word hypothesis must start at the end of a previous hypothesis. Not shown in this figure are the acoustic and language model probabilities that decorate each arc.



$$H^*(s) = f(s) + g^*(s)$$

$H^*(s)$ 是经过状态s，最好的完整路径分数。

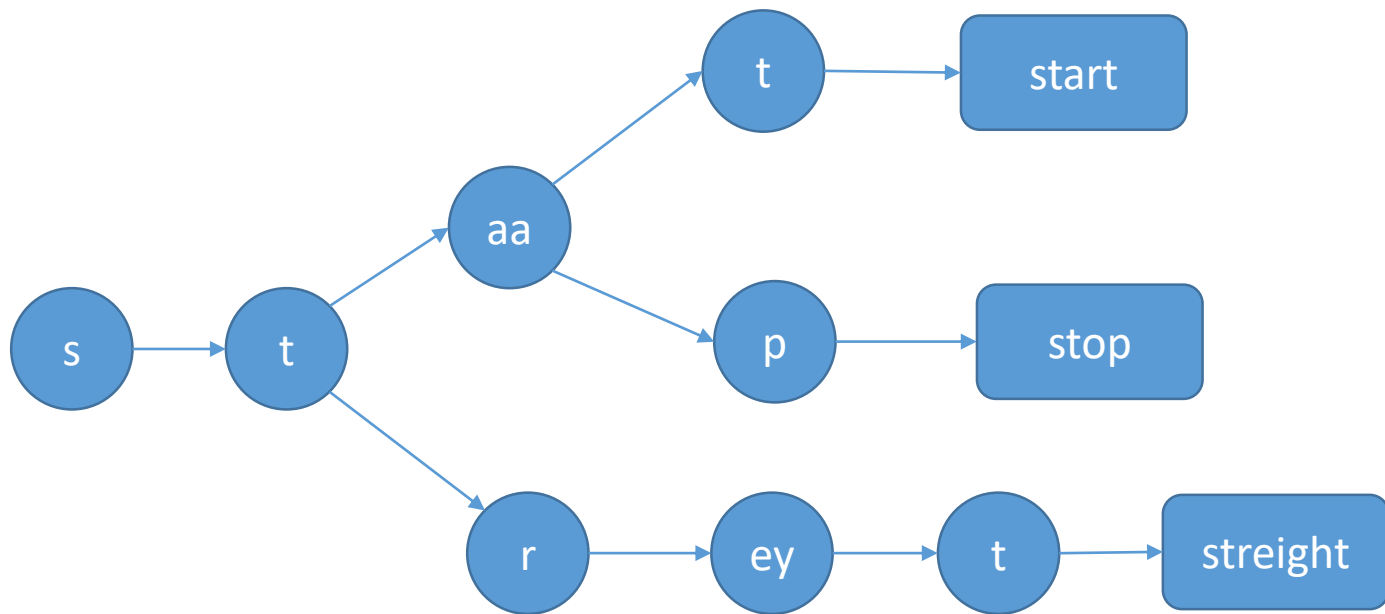
$f(s)$ 是从初始到状态s的部分路径分数(真实)。

$g^*(s)$ 是估计的从状态s到结束这个部分路径的最优分数(估计)。

如何能提出好的 $g^*(s)$ 估计至今
是研究者们探究的问题。



解码器—树状字典





在Tree Structured decoder中，把LM概率分配到结点上，而不是走到叶子结点才累积LM概率，从而更早的剪枝。

$$P(j|i, h) = \frac{\max_{w \in \sigma(j)} P(w|h)}{\max_{v \in \sigma(i)} P(v|h)}$$

在一个字典树上，N-gram的history是h，i和j为树上两个结点，w为从j结点能到达的所有词，v为从i结点能到达的所有词。那么从i结点到j结点上的概率由上式计算。



试想你用高阶N-gram语言模型，一个有数以百万词的字典，构建一个解码图。



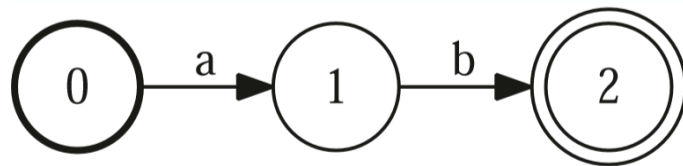
冗杂



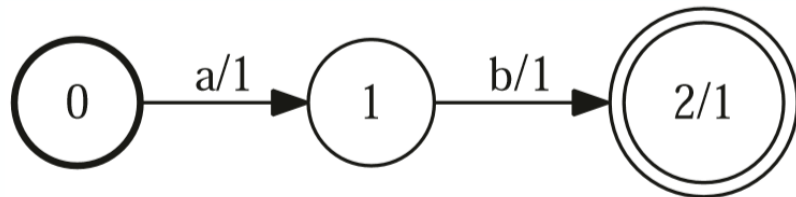
高效，统一的解决办法：WFST构图



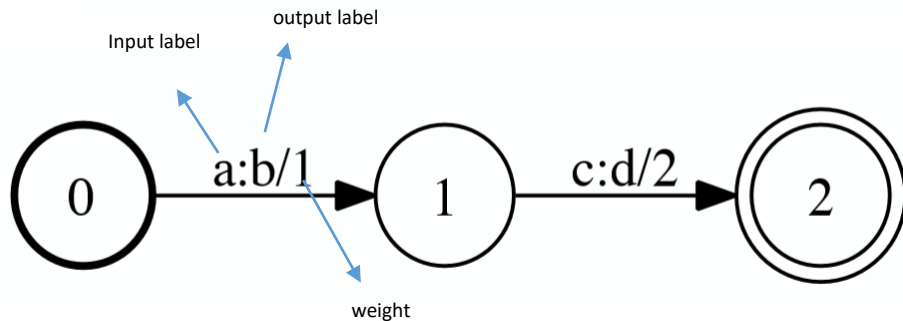
有限状态接收器(FSA): Finite-state Acceptor



加权有限状态接收器(WFSA):
Weighted Finite-state Acceptor



加权有限状态转换器(WFST):
Weighted Finite-state Transducer:





WFST介绍—半环

半环:

- 有一个元素集合(e.g. R)
- 有两个特殊元素 $\bar{0}$ (零元)和 $\bar{1}$ (幺元)
- 有两个操作 \oplus (加操作)和 \otimes (乘操作)。
 - 加操作有交换律, 结合律, 与零元相加为本身
 - 乘操作有结合律, 于幺元乘为本身
 - 分配律: $w_1 \otimes (w_2 \oplus w_3) = (w_1 \otimes w_2) \oplus (w_1 \otimes w_3)$
 - 任意数与零元乘操作为零元。

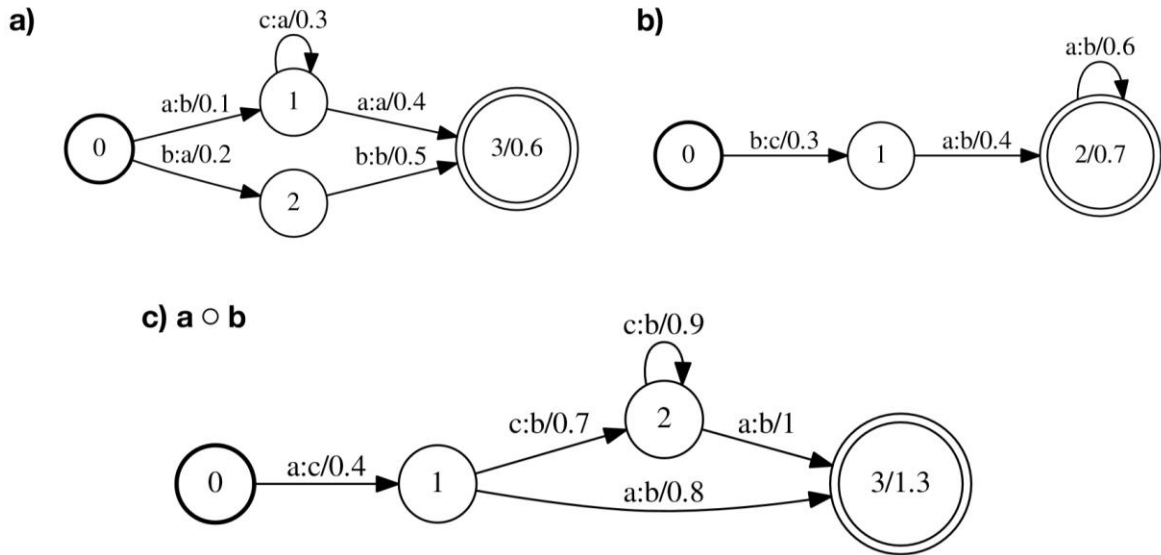
SEMIRING	SET	\oplus	\otimes	$\bar{0}$	$\bar{1}$
Boolean	$\{0, 1\}$	\vee	\wedge	0	1
Probability	\mathbb{R}_+	+	\times	0	1
Log	$\mathbb{R} \cup \{-\infty, +\infty\}$	\oplus_{\log}	+	$+\infty$	0
Tropical	$\mathbb{R} \cup \{-\infty, +\infty\}$	min	+	$+\infty$	0

From Mehryar Mohri, Speech Recognition with WFST



WFST介绍--Composition

组合：如果一个转换器A将序列x映射到序列y伴随着权重a，并且转换器B将序列y映射到序列z伴随着权重b，那么组合的转换器将序列x映射到序列z，权重为 $a \otimes b$ 。



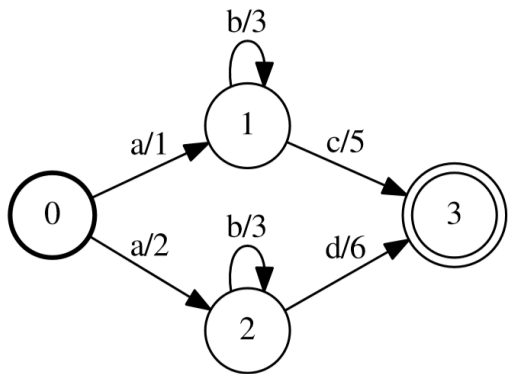
假设Tropical半环



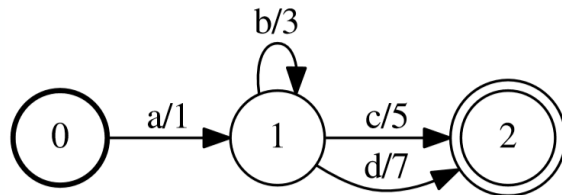
WFST介绍--Determinization

确定化：创建等价的FST，任意一个状态都没有两个相同input label的出弧(arc).
条件：这个FST是functional的，即每一个输入序列可以转换成**独一无二**的输出序列。

FSA A)



A')

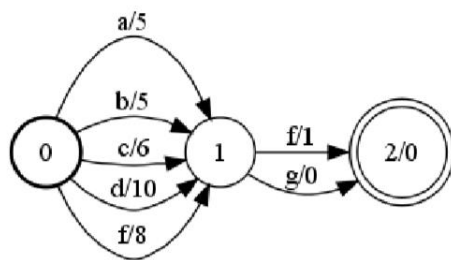
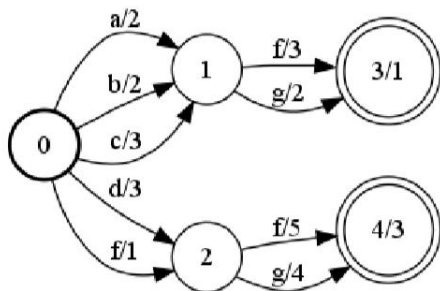




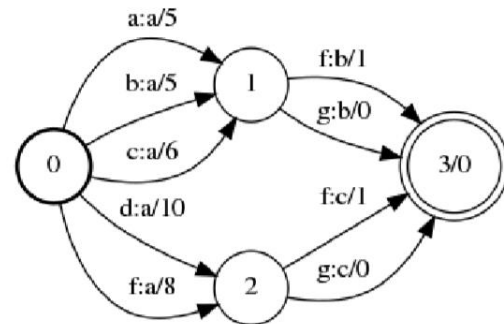
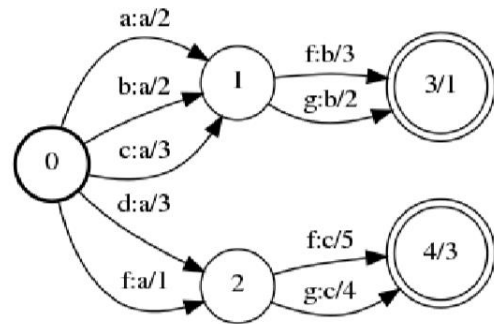
WFST介绍--Minimization

最小化：创建等价的FST，拥有最少的状态数和弧数。

WFSA



WFST





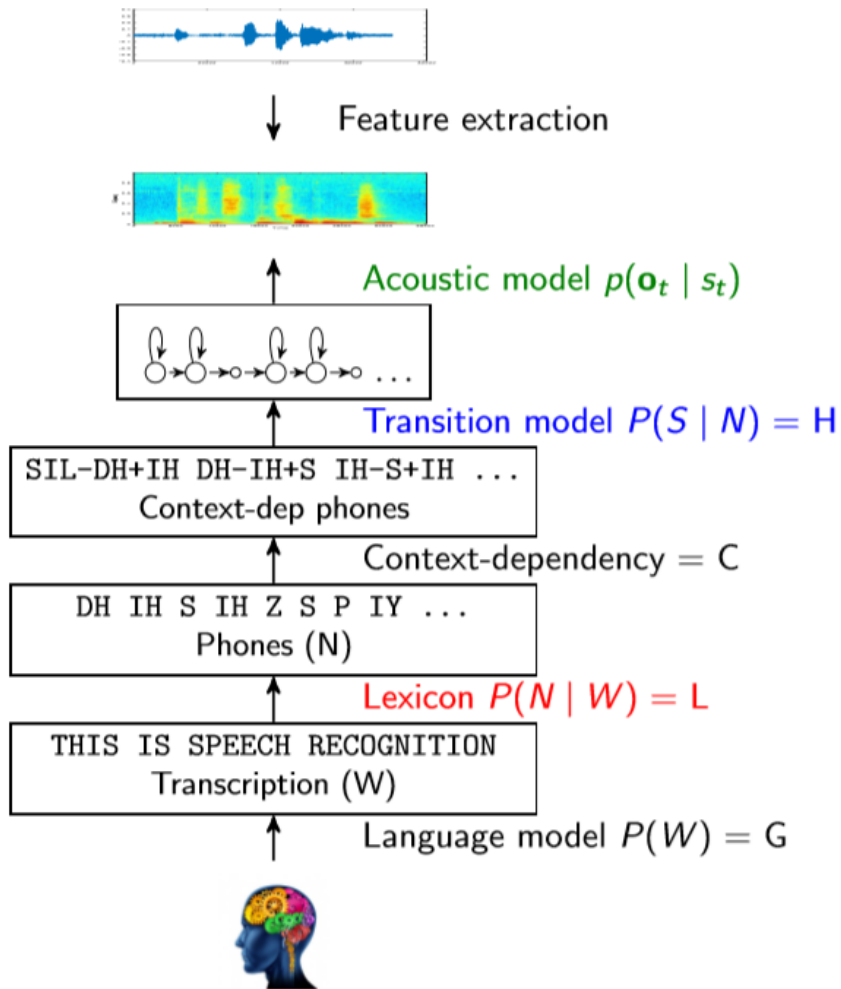
WFST介绍—其它操作

- 弧排序(ArcSort):根据input label或者output label排序每个state的arcs。
- 链接(Connect): 剪枝FST, 去掉所有不在成功路径(从初始状态到终止状态) 的state和arcs。
- 相等(Equal): 确定两个FST(A和B)有相同数量和顺序的state, arcs。
- 等价(Equivalent):确定两个不含epsilon的确定化状态机(A和B)等价, 即对于相同的输入, 有相同输出和权重。
- 推(Push): 将权重向初始状态或者终止状态推动。
- . . .

小贴士: 多看
openfst官网



识别中的WFST

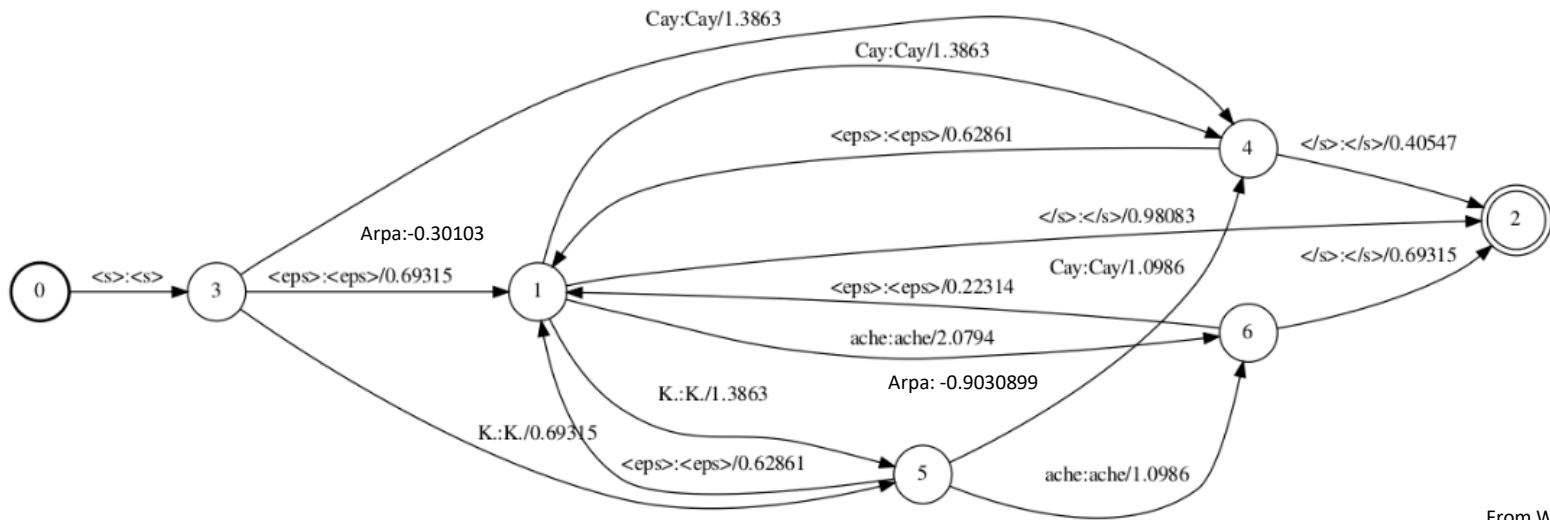




G.Fst示例解析

State 0和1为history=空

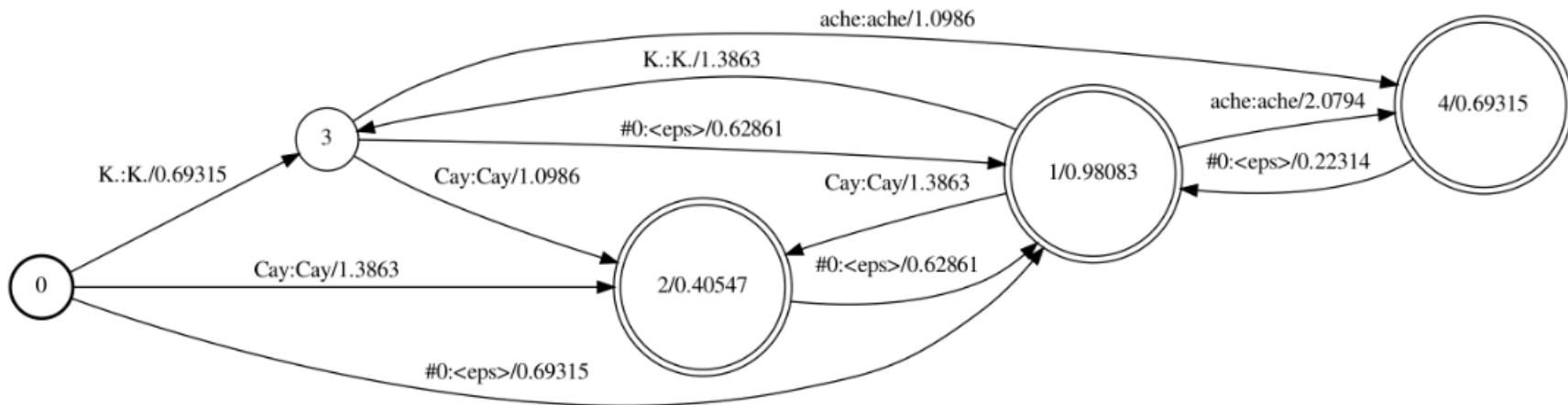
State 2-6为history=</s>,<s>,<Cay>,<K>,<ache>





确定化的G.fst

- a. 用#0替换backoff边的input label
- b. 用epsilon替换<s>和</s>
- c. 确定化





L.Fst介绍:

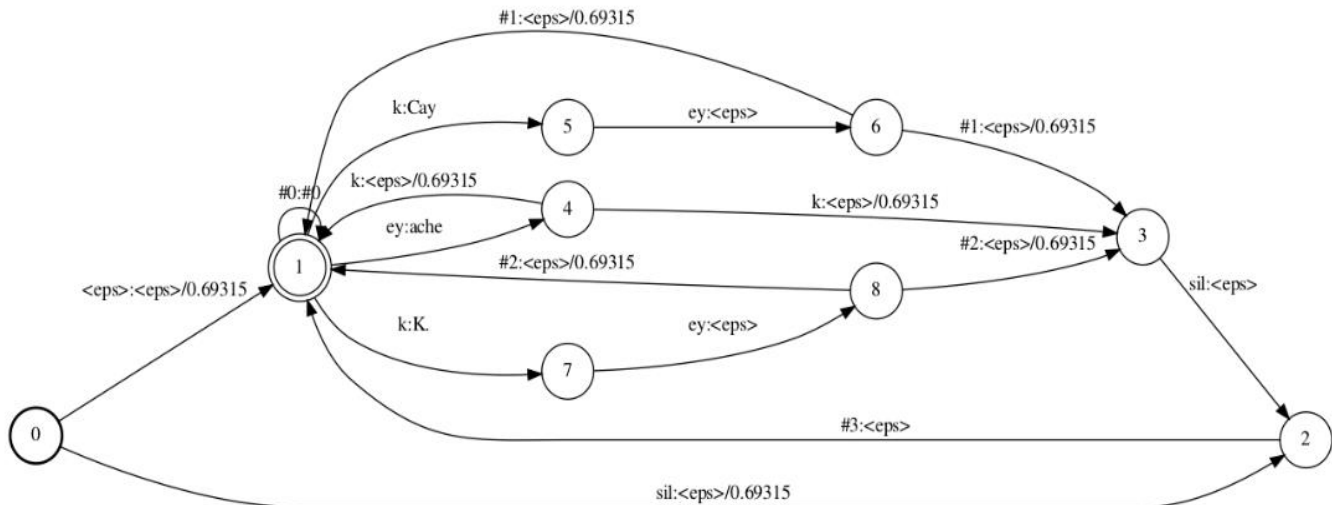
- a. 消歧义符(disambiguation symbol,#1,#2...):解决发音前缀和同音异形字。
- b. 为词前后添加silence。
- c. Add-self-loop:为终止状态添加#0的自环,从而和G.fst合并。

尝试自己写出字典:

Cay k ey

ache ey k

K. K ey



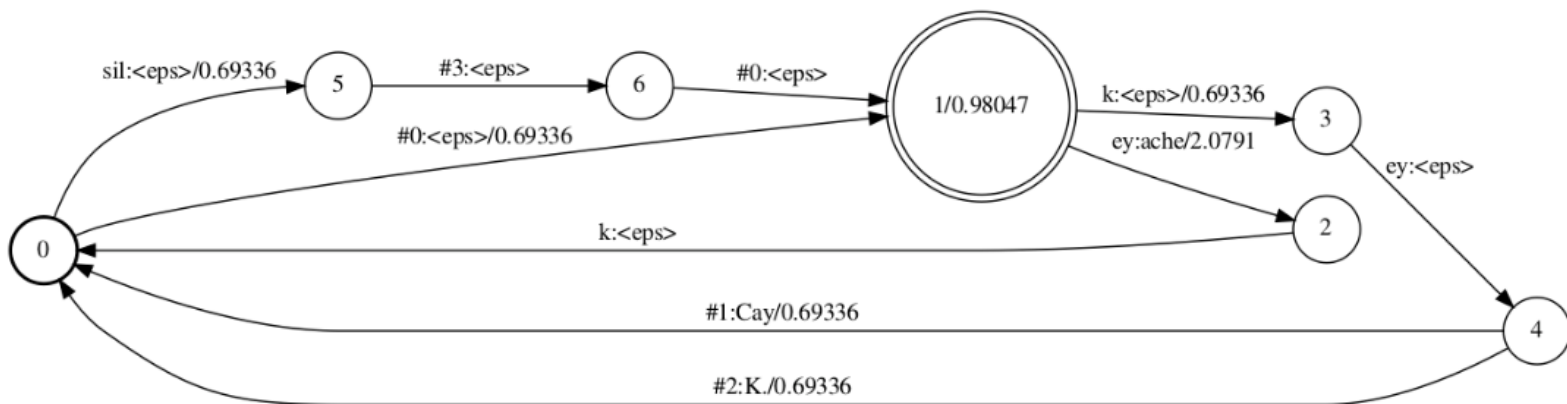


识别中的WFST

L compose G:

[addsubsequentialloop由于C的尾处理]

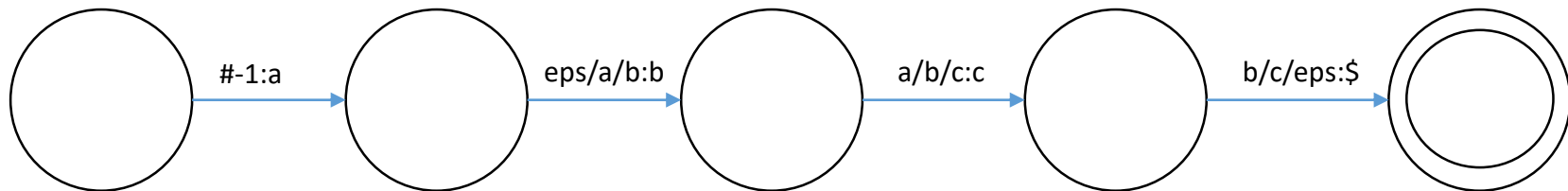
Kaldi有一些自己的fst command-line, 略不同于Openfst, 源于具体问题的处理。使用fst时善用openfst/bin, openfst/src, kaldi/src/fstbin, kaldi/src/fstext





- a. 通常Kaldi里不单独生成C，而是直接与LG进行compose，生成CLG [fstcomposecontext]。这样可以动态生成，避免穷举所有cd-phone.
- b. Kaldi用N表示窗长，P表示中心音素位置。[left-context1/phone/right-context1]=[N=3,P=1]
- c. (N=3,P=1为例):每个arc的格式为left/phone/right:right，如a/b/c:c，这里输出的不是中心音素。
- d. 用#-1和\$处理开头结尾。
- e. 决策树会将你想的逻辑cd-phone变为绑定后的形式。[make-ilabel-transducer]

Logical cd-phone C的示意图: [Morhi的表示方法不同,格式为phone:phone/left_right]





H Fst:

理想化：我们只要把pdf-id到cd-phone就可以了。

但由于kaldi的决策树一个pdf-id可以对应若干cd-phone,所以引入了

transition-id = (transition-state, transition-index)

transition-state = (phone, hmm-state, forward pdf, self-loop pdf)—new

= (phone, hmm-state, pdf)—new

$$HCLG = asl(min(rds(det(H_a \circ min(det(C \circ min(det(L \circ G))))))))$$



加自环



去除消歧义符号

Kaldi

$$HCLG = rds(min(det(H \circ det(C \circ det(L \circ G))))))$$

Mohri.



识别中的WFST

	Input	output
H (HMM)	HMM状态 (transition-id kaldi)	上下文相关音素
C (Context-Dependency)	上下文相关音素	音素
L (Lexicon)	音素	词
G (grammar/language model,acceptor)	词	词

对于HCLG: 每个arc的ilabel=transition-id, olabel=word-id, weight为transition概率,LM概率等, weight pushing后的值。

有了解码图,
逐帧去搜索吧!



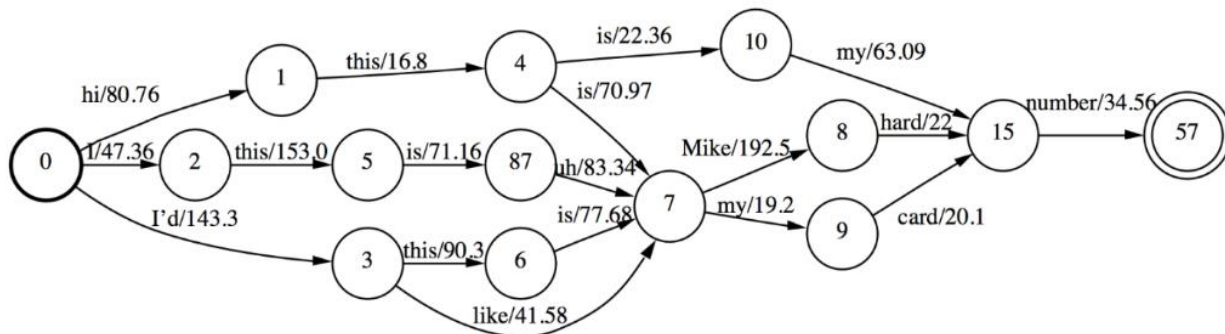
识别中的WFST—再看lattice

对于HCLG：每个arc的ilabel=transition-id, olabel=word-id, weight。

在kaldi里有Lattice和CompactLattice，是一个事物的两种存储形式，可以相互转化，在外观上都和HCLG相似，由state和arc组成，从中你可以知道概率分数和时间。

Lattice的arc: ilabel=transition-id, olabel=word-id, weight=(graph_cost, acoustic_cost)

CompactLattice的arc: ilabel=olabel=word-id, weight=(graph_cost, acoustic_cost, transition-id sequence).





结语

感谢各位聆听!

Thanks for Listening

