

Lecture 8: 统计语言模型

Kai Yu and Yanmin Qian

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

Spring 2021



- ▶ 语音识别中的语言模型
 - ▶ 语法 - Grammar
 - ▶ N 元文法 - N-gram
- ▶ N-gram 语言模型
- ▶ 评估
- ▶ Discounting and back-off
- ▶ 语言模型在语音识别中的应用



语言模型建模

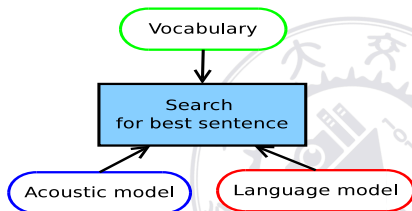
概念

$$\hat{W} = \arg \max_W P(W|\mathbf{O}) = \arg \max_W \underbrace{p(\mathbf{O}|W)}_{\text{acoustic model}} \underbrace{P(W)}_{\text{language model}}$$

其中 $W = [w_1, \dots, w_N]$ 通常情况下表示词序列。

语言模型建模需要定义一个词表。搜索空间随着词表的大小会呈指数级的增长。那些在词表中无法找到的词称为集外词

Out-Of-Vocabulary (OOV) words。期望是，尽量保持 OOV 尽可能的低。



语言模型建模

为什么在语音识别中很重要

那些具有相同发音的句子可能不都是有意义的：

英文句示例


- ▶ I went to a party. (✓)
- ▶ Eye went two a bar tea.

中文句示例

- ▶ 你 现在 在 干什么？(✓)
- ▶ 你 西安 载 感 什么？

语言模型建模

为什么在语音识别中很重要



我们的家园

STUDENT COMMUNITY OF SHANGHAI JIAO TONG UNIVERSITY

我们的家园-微意见

紫荆园早餐卖隔夜粥

2021.04.21 07:29:53来源:

今天早上去紫荆园最左边包子的那个窗口打餐,要一份八宝粥,工作人员提示说八宝粥是隔夜的,问我可不可以。百年名校的食堂卖隔夜的粥,不怕出现卫生事件吗。

答复信息如下:

Re:紫荆园早餐卖隔夜粥

饮食服务中心 发布于2021.04.21 09:06

同学你好,同学反映的问题食堂很重视。食堂菜品均为当天制作当天出售,严格保证食品安全。同学反映的问题我们已核实。该窗口售卖有荷叶粥菜品,并无八宝粥。工作人员是在征询同学意见,荷叶粥是否可以,还请同学知悉。在就餐过程中同学遇到问题可第一时间与食堂值班经理联系 () 或1 (),以便我们能及时为同学们解决问题,为同学们提供更好的餐饮服务。感谢同学的宝贵意见,欢迎同学继续监督我们的工作。饮食中心紫荆园食堂

回复7条

183.173.140.155 回复 2021年04月21日 10:54
啊发发发发发.....

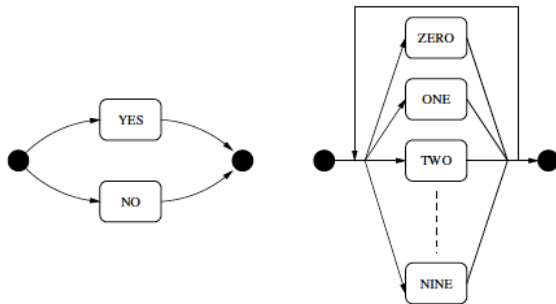
183.173.54.11 回复 2021年04月21日 10:53
打卡,笑yue了

183.173.112.123 回复 2021年04月21日 10:51
打卡

183.173.78.232 回复 2021年04月21日 10:47
哈哈哈哈哈 笑抽了 谢谢这位同学 这是我今天的快乐源泉

阿拉好宁波

简单的语法



- ▶ 一个语法网络是一个加权有限状态机 **WFSM**
- ▶ 语法包括词之间的回环，替代，以及重复等。
- ▶ 对那些经常观测到的路径，可以在对应弧 (arcs) 上赋予一个较高分数的权重概率。

统计语言模型

语法网络的使用有很多限制

- ▶ 很难手工去生成大量的语法规则
- ▶ 自然语音的说话通常情况下是不合语法的

统计语言模型是一个很吸引人的可替代方法

- ▶ 估计过程是完全基于数据驱动的 (no handcraft)
- ▶ 适合自然语音

< sent_start > the overall operation ... string < sent_end >
 w_0 w_1 w_2 w_3 ... w_K w_{K+1}

$$P(W) = P(w_1, w_2, \dots, w_K, w_{K+1}) = \prod_{k=1}^{K+1} P(w_k | w_1, \dots, w_{k-1})$$

句子结尾标识的使用允许实现对句子结束的预测。

N-gram 语言模型

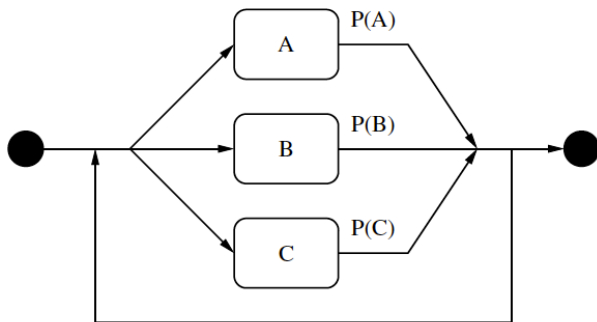
N-gram 模型近似地表示了在一个较短历史情况下的条件概率 (up to N)

$$P(w_k | w_1, \dots, w_{k-1}) = P(w_k | w_{k-1}, \dots, w_{k-n+1})$$

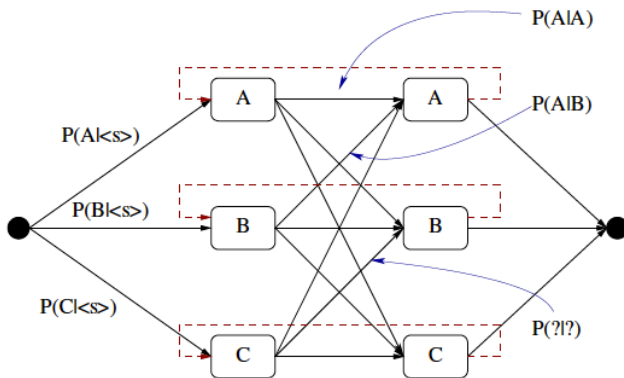
给定的一个当前词的概率仅仅依赖于这个词之前的 $n - 1$ 个前继词。通常使用的 N-gram 模型是

- ▶ $N = 1$: unigram
 - ▶ $N = 2$: bigram
 - ▶ $N = 3$: trigram
 - ▶ $N = 4$: 4-gram(quadrigram)
- ▶ $\hat{P}(\omega_\kappa)$
 - ▶ $\hat{P}(\omega_\kappa | \omega_{\kappa-1})$
 - ▶ $\hat{P}(\omega_\kappa | \omega_{\kappa-2}, \omega_{\kappa-1})$
 - ▶ $\hat{P}(\omega_\kappa | \omega_{\kappa-3}, \omega_{\kappa-2}, \omega_{\kappa-1})$

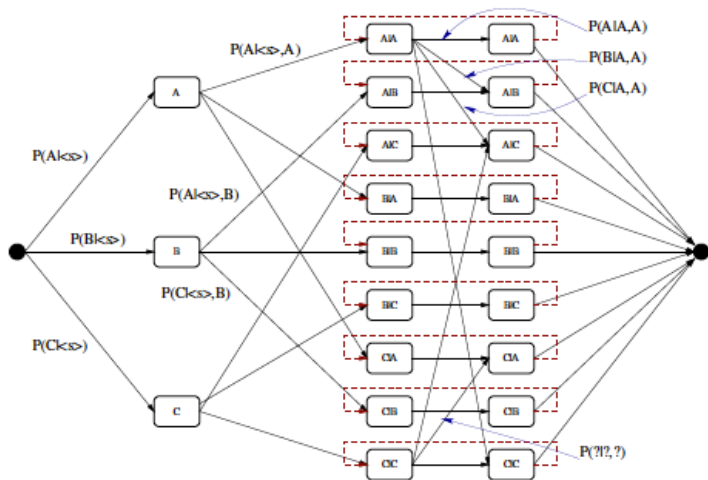
Unigram Language Model



Bigram Language Model



Trigram Language Model



文本归一化的目标是转换训练文本，使得它变成纯语言的形式，同时移除很多和语言完全不相关的部分。典型的文本归一化包括：

- ▶ 标点的移除：
 - ▶ +@()!,' +
- ▶ 日期，货币，数字的归一化：
 - ▶ 10/4/1998 \Rightarrow 1998 年 10 月 4 日
 - ▶ \$12.50 \Rightarrow 12.5 美元
 - ▶ 3.67% \Rightarrow 百分之 3.67%
- ▶ 缩写词的归一化：
 - ▶ CNN \Rightarrow C. N. N.



N-gram 语言模型的估计

最大似然准则

数据: 词序列 $W_1^N = [w_1, \dots, w_N]$

模型: 离散概率

$$P(w_k | W_{k-n+1}^{k-1}), \quad w \in \mathcal{V}$$

其中 $\mathcal{V} = \{v_1, \dots, v_M\}$ 是词表

Criterion:

$$\begin{aligned}\mathcal{L}(\theta) &= \frac{1}{K} \sum_{k=1}^K \log P(w_k | W_{k-n+1}^{k-1}) \\ &= \frac{1}{K} \sum_{v \in \mathcal{V}} \sum_{y \in \mathcal{Y}} \sum_{k: w_k=v, y=W_{k-n+1}^{k-1}} \log P(v|y) \\ &= \frac{1}{K} \sum_{v \in \mathcal{V}} \sum_{y \in \mathcal{Y}} C(y, v) \log P(v|y)\end{aligned}$$

其中 \mathcal{Y} 是所有可能的 N-gram 历史的集合, $C(y, v)$ 是训练集中完整 N-gram (y, v) 的数目。

N-gram Language Model 的估计

优化

$$\hat{P}(v|y) = \arg \max_{P(v|y)} \sum_{v \in \mathcal{V}} \sum_{y \in \mathcal{Y}} C(y, v) \log P(v|y), \quad s.t. \sum_{v \in \mathcal{V}} P(v|y) = 1$$

使用拉格朗日乘子，针对参数 $P(v|y)$ 的带约束的优化可以表示为

$$Q(P(v|y)) = C(y, v) \log P(v|y) + \lambda \left(\sum_{v \in \mathcal{V}} P(v|y) - 1 \right), v \in \mathcal{V}$$

分别关于参数 $P(v|y)$ 与 λ 对以上式子求偏导数，并令其等于零：

$$\frac{\partial Q}{\partial P(v|y)} = 0, v \in \mathcal{V} \qquad \frac{\partial Q}{\partial \lambda} = 0$$

N-gram Language Model 的估计

优化

$$\begin{aligned} \frac{C(y, v)}{\hat{P}(v|y)} + \lambda &= 0, v \in \mathcal{V} \\ -\frac{C(y, v)}{\lambda} &= \hat{P}(v|y), v \in \mathcal{V} \end{aligned} \quad \Rightarrow \quad \begin{aligned} \sum_{v \in \mathcal{V}} \hat{P}(v|y) - 1 &= 0 \\ -\frac{\sum_{v \in \mathcal{V}} C(y, v)}{\lambda} &= 1 \\ -\sum_{v \in \mathcal{V}} C(y, v) &= \lambda \end{aligned}$$

因此, 对 $\forall v \in \mathcal{V}$, 有

$$\hat{P}(v|y) = \frac{C(y, v)}{C(y)}$$

其中

$$C(y) = \sum_{v \in \mathcal{V}} C(y, v)$$

N-gram Language Model 的估计

Unigram 示例

Training sentences:

| |
|-------------------------------|
| <S> I HAVE A RED CAR </S> |
| <S> I BUY A NEW CAR </S> |
| <S> THEY HAVE A NEW BOOK </S> |

Vocabulary:

| | | | | | | | | |
|---|------|-----|-----|------|---|-----|-----|------|
| A | BOOK | BUY | CAR | HAVE | I | NEW | RED | THEY |
|---|------|-----|-----|------|---|-----|-----|------|

Unigram language model:

| | | | | | |
|----------|------|-----------|------|-----------|------|
| $P(A)$ | 3/15 | $P(BOOK)$ | 1/15 | $P(BUY)$ | 1/15 |
| $P(CAR)$ | 2/15 | $P(HAVE)$ | 2/15 | $P(I)$ | 2/15 |
| $P(NEW)$ | 2/15 | $P(RED)$ | 1/15 | $P(THEY)$ | 1/15 |

N-gram Language Model 的估计

Bigram 示例

Training sentences:

<S> I HAVE A RED CAR </S>
<S> I BUY A NEW CAR </S>
<S> THEY HAVE A NEW BOOK </S>

Bigram language model:

| | | | | | |
|---------------|-----|------------------|-----|----------------|-----|
| $P(I <S>)$ | 2/3 | $P(THHEY <S>)$ | 1/3 | $P(NEW A)$ | 2/3 |
| $P(RED A)$ | 1/3 | $P(A BUY)$ | 1/1 | $P(A HAVE)$ | 2/2 |
| $P(HAVE I)$ | 1/2 | $P(BUY I)$ | 1/2 | $P(CAR NEW)$ | 1/2 |
| $P(BOOK NEW)$ | 1/2 | $P(CAR RED)$ | 1/1 | $P(HAVE THEY)$ | 1/1 |

如何测量 LM 的质量

交叉熵 – Cross Entropy

语言模型的评价方法：

- ▶ 理想中的：直接语音识别的结果 – 词错误率（WER）（使用相同的声学模型）
- ▶ 实际中的：在预留的文本数据集上的预测度

Cross-entropy：在 N-gram 模型 $P(\cdot)$ 和实际的开发文本集合上的实际词分布 $P_{\text{data}}(\cdot)$ 之间的 cross-entropy 被广泛地用作 LM 的评价指标。可以认为它是一个平均对数（底数为 2）似然分数。

$$\begin{aligned} H(P, Q) &= - \sum_{v \in \mathcal{V}} \sum_{y \in \mathcal{Y}} P_{\text{data}}(v|y) \log_2 P(v|y) \\ &\approx - \frac{1}{K} \sum_{v \in \mathcal{V}} \sum_{y \in \mathcal{Y}} C(y, v) \log_2 P(v|y) \\ &\approx - \frac{1}{K} \sum_{k=1}^K \log_2 P(w_k | W_{k-n+1}^{k-1}) = -\mathcal{L}(\theta) \end{aligned}$$

如何测量 LM 的质量

困惑度 – Perplexity

Perplexity 定义如下

$$\begin{aligned}\text{PPL} &= 2^{-\mathcal{L}(\theta)} = 2^{-\frac{1}{K} \sum_{k=1}^K \log_2 P(w_k | W_{k-n+1}^{k-1})} \\ &= \left(2^{-\sum_{k=1}^K \log_2 P(w_k | W_{k-n+1}^{k-1})} \right)^{\frac{1}{K}} = \left(\prod_{k=1}^K 2^{-\log_2 P(w_k | W_{k-n+1}^{k-1})} \right)^{\frac{1}{K}} \\ &= \left(\prod_{k=1}^K \frac{1}{P(w_k | W_{k-n+1}^{k-1})} \right)^{\frac{1}{K}}\end{aligned}$$

- ▶ Perplexity 是 N-gram 概率倒数的几何平均
- ▶ 可以被理解或解释成**分支度 branching factor**
- ▶ 越低的 PPL 说明越好的预测能力

Perplexity 示例

电话号码 的例子。假定

- ▶ 所有电话号码都是 6 位数字长度
- ▶ 所有数字都是等概率的以"33", "74", "76" 开头
- ▶ 所有其他的数字都是等概率的

实验中, 使用 10000 或者 100 个数字串去训练 N-gram 语言模型, 然后使用 1000 数字串去测试。

| Language Model | 10000 | | 100 | |
|----------------|-------|-------|-------|----------|
| | Train | Test | Train | Test |
| Equal | 11.00 | 11.00 | 11.00 | 11.00 |
| Unigram | 10.04 | 10.01 | 10.04 | 10.04 |
| Bigram | 7.12 | 7.13 | 6.56 | ∞ |

Q: 为什么 bigram 模型的 perplexity 测试可能是 ∞ ?

PPL 和 WER 之间的关系

- ▶ 总体规则: 随着 perplexity (PPL) 的降低, 语音识别的 word error rate (WER) 也会降低
- ▶ **Rule-of-thumb:** 给定相同类型下的语言模型

$$\text{WER} \approx \kappa \sqrt{\text{PPL}}$$

$$\text{WER} \approx \kappa \log(\text{PPL})$$

其中 κ 是任务相关的

- ▶ 上述 *Rule-of-thumb* 很大程度上受词表大小, 语言种类, 以及任务的影响。

N-gram LM 的数据稀疏问题

数据稀疏问题一直存在：大词表情况下的高阶 N-gram。数据稀疏的相关问题主要有如下：

- ▶ **Unseen** 零概率的 N-grams 将会导致无穷大的分支度 perplexity
- ▶ **Very low** 低频次的 N-grams 将会导致不可靠的估计

两种解决方案：

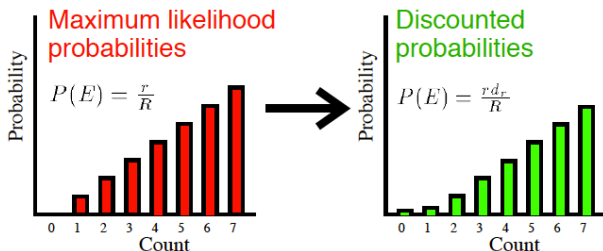
- ▶ **Discounting:**

Discounting 折扣法修正原始 N-gram 的频次，使得概率量重新分布，从通常容易观测到的 N-gram 分布 \rightarrow 频次较低或者根本没有的 N-gram 上。

- ▶ **Backing-off and interpolation:**

Back-off 回退策略递归地回退到较低阶的 N-gram 上，直到得到一个鲁棒的概率估计。

Discounting



为了需要重新分布一些概率值到一些 *unseen* 没有看到过的 N-gram 上，我们需要去折扣（减少）那些在原本训练文本中的看到过的 N-gram 的频次，从而来满足概率和为 1 的要求。

$$\hat{P}_{\text{ML}}(w|y) = \frac{C(y, w)}{C(y)} \quad \hat{P}_{\text{Discount}}(w|y) = d(y, w) \frac{C(y, w)}{C(y)}$$

其中 $d(y, w)$ 是一个折扣 (discount) 系数

Good Turing Discounting

基本的想法：将训练数据中出现次数为 $r + 1$ 次的 N-gram 的概率重新分配给出现次数为 r 次的 N-gram。

$$r^* = (r + 1) \frac{N_{r+1}}{N_r} \quad d(r) = \frac{(r + 1)N_{r+1}}{rN_r}$$

其中 N_r 表示 **count of counts**, i.e. 在训练集中出现次数是 r 次的 N-gram 的数量。因而，训练集中所能观测到的 N-grams 的总数是：

$$N = \sum_{r=0}^{\infty} rN_r = \sum_{r=1}^{\infty} rN_r$$

一个 N-gram 的概率可以定义为

$$P(v : C(v) = r) = \frac{r^*}{N}$$

- ▶ 在进行折扣算法之前，需要先对 N_r 做一些调整
- ▶ 分配给 unseen 的 N-gram 上的概率量是 $\frac{N_1}{N}$



Good Turing Discounting

示例

Given the following counts:

| Symbol | A | B | C | D | E | F | G | H | I | J |
|--------|---|---|---|---|---|---|---|---|---|---|
| Counts | 1 | 1 | 0 | 2 | 1 | 2 | 2 | 0 | 1 | 3 |

Frequency table: (applying $r^* = (r + 1) \frac{N_r + 1}{N_r}$)

| r | Symbols | N_r | rN_r | $P(x)$ |
|-------|------------|-------|--------|--------|
| 0 | C, H | 2 | 0 | 0 |
| 1 | A, B, E, I | 4 | 4 | 4/13 |
| 2 | D, F, G | 3 | 6 | 6/13 |
| 3 | J | 1 | 3 | 3/13 |
| Total | | 13 | | 1 |

→

| r^* | Symbols | N_r | r^*N_r | $P(x)$ |
|-------|------------|-------|----------|--------|
| 2 | C, H | 2 | 4 | 4/13 |
| 3/2 | A, B, E, I | 4 | 6 | 6/13 |
| 1 | D, F, G | 3 | 3 | 3/13 |
| 1 | J | 1 | 0 | 0 |
| Total | | 13 | | 1 |

- ▶ 那些出现 r 次的 N-gram 的概率已被重新分配给那些出现了 $r + 1$ 次的 N-gram。
- ▶ Good turing discounting 通常仅仅应用于低频次的词

Jelinek-Mercer Smoothing (Interpolation)

Good-Turing discounting: unseen N-grams 被赋予相同的频次

$$\left. \begin{array}{l} C(\text{Be}, \text{Happy}) = 0 \\ C(\text{Be}, \text{ZZZ}) = 0 \end{array} \right\} \Rightarrow P(\text{ZZZ}|\text{Be}) = P(\text{Happy}|\text{Be})$$

这个看来好像是有问题的，因为 $P(\text{Happy}) > P(\text{ZZZ})$

解决方案：插值 在 unigram 和 bigram 之间

$$P^*(w_i|w_{i-1}) = \lambda P(w_i|w_{i-1}) + (1 - \lambda)P(w_i)$$

这个可以推广到高阶的情况

$$P^*(w_i|W_{i-n+1}^{i-1}) = \lambda_n P(w_i|W_{i-n+1}^{i-1}) + (1 - \lambda_n)P(w_i|W_{i-n+2}^{i-1})$$

λ_n 可以在预留出来的开发数据集上学习得到。

Katz's Back-off Model

回退 **Back-off**: 在估计那些训练文本中从没见过 (unseen) 或者见过频次很少 (rarely seen) 的 N-gram 的概率时, 使用某些条件下相对较小的历史。

$$P_{\text{bo}}(w_k | W_{k-n+1}^{k-1}) = \begin{cases} d(W_{k-n+1}^k) \frac{C(W_{k-n+1}^k)}{C(W_{k-n+1}^{k-1})} & C(W_{k-n+1}^k) > t \\ \alpha(W_{k-n+1}^{k-1}) P_{\text{bo}}(w_k | W_{k-n+2}^{k-1}) & \text{otherwise} \end{cases}$$

- ▶ t 一般情况下置成零 (back-off only if N-gram is unseen)
- ▶ d 和 α 分别是折扣因子 (discounting factor) 和回退权重 (back-off weights)
- ▶ Good-Turing discounting 很广泛地被使用
- ▶ 选取 α 使得

$$\sum_{w_k}^V P_{\text{bo}}(w_k | W_{k-n+1}^{k-1}) = 1$$

语言模型插值

- ▶ 对同一个语言模型中的不同阶数 N-gram 的概率进行插值
- ▶ 对具有相同 **N-gram** 阶数的不同语言模型进行插值

$$P(w_i|W_{i-n+1}^{i-1}) = \sum_{m=1}^M \lambda_m P_m(w_i|W_{i-n+1}^{i-1})$$

其中 λ_m 可以通过在一个预留的开发集合上通过 MLE 最大似然准则估计得到

- ▶ 对综合来自不同数据源的信息很有用
- ▶ 做 LM 自适应的快速方式

Class-based Language Model

- ▶ Class-based LM 使用参数绑定的方法来得到更加鲁棒的 N-gram 概率估计
- ▶ 将 N-gram 的历史信息归成不同的类别

$$P(w_i | W_{i-n+1}^{i-1}) = P(w_i | G_i) P(G_i | G_{i-n+1}^{i-1})$$

其中 $G(w)$ 表示映射词 w_i 为它的相关类或组的函数。

I (noun) buy (verb) a (article) book (noun).

He (noun) sold (verb) a (article) car (noun).

- ▶ 归类或分组工作一般在词层面来实现
- ▶ Class-based LMs 往往和 word-based LMs 进行插值来得到进一步的性能改进

语言模型在语音识别中的应用

将语言模型集成至语音识别系统中通常有 2 种策略：**First-Pass** 与 **Second-Pass**。

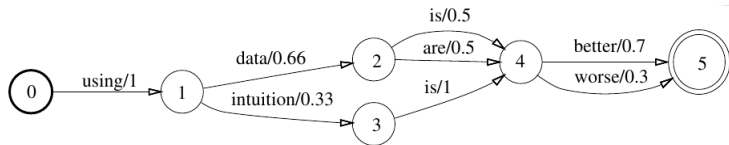
- ▶ **First-Pass**：在识别解码过程中引入语言模型打分，作为选择最优解码路径的依据之一。
 - ▶ 该策略能够及时对不合自然语言规则的解码路径进行剪枝，但若语言模型过大，将显著影响解码效率。
- ▶ **Second-Pass**：在解码得到若干可能结果序列后，使用语言模型对每个序列重打分（rescore），并选择得分最高的序列作为最终识别结果。
 - ▶ 该策略对识别效率的影响更为可控，可以集成更大的语言模型，但系统识别性能严重依赖于初始结果序列集合的质量。

实际系统中，通常同时使用两种策略（即两趟解码）：将一个小的语言模型集成至解码过程中帮助剪枝，并使用另一个更大的语言模型进行重打分。

加权有限状态转录机 (WFST)

加权有限状态转录机 (Weighted Finite-State Transducer, WFST) 是语音识别构图的一种主流方法。它由一组状态 (State) 和状态间的有向跳转 (Transition) 构成, 每个跳转上包含 3 种信息: 输入标签, 输出标签和权重。

有关 WFST 的详细介绍, 可以参考 Mohri, et al. *Speech Recognition With Weighted Finite-State Transducers*.



使用 OpenFst 构建语法规则网络

Kaldi 的 WFST 实现基于 OpenFst。OpenFst 定义了一种用于描述 WFST 的语言，提供了 WFST 的表示和各种操作的实现。

fst.txt

```
0 1 using      using      1
1 2 data       data       0.66
1 3 intuition  intuition  0.33
2 4 is         is         0.5
2 4 are        are        0.5
3 4 is         is         1
4 5 better     better     0.7
4 5 worse      worse      0.3
5 1.0
```

symbols.txt

```
<eps> 0
using 1
data 2
intuition 3
is 4
are 5
better 6
worse 7
```

```
fstcompile --isymbols=symbols.txt \
```

```
--osymbols=symbols.txt fst.txt > out.fst
```