语言模型
（Language Model, LM）

洪青阳 副教授

厦门大学信息科学与技术学院
qyhong@xmu.edu.cn

要点

- 语言模型背景
- N-gram语言模型
- 评价指标
- 平滑技术
- 训练工具

语音识别任务

- 语音识别的任务为，找到对应观察序列O的最可能的词序列$\hat{W}$。按贝叶斯准则：

$$\hat{W} = arg\max_{W} P(W|O) = arg\max \frac{P(W)P(O|W)}{P(O)}$$

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

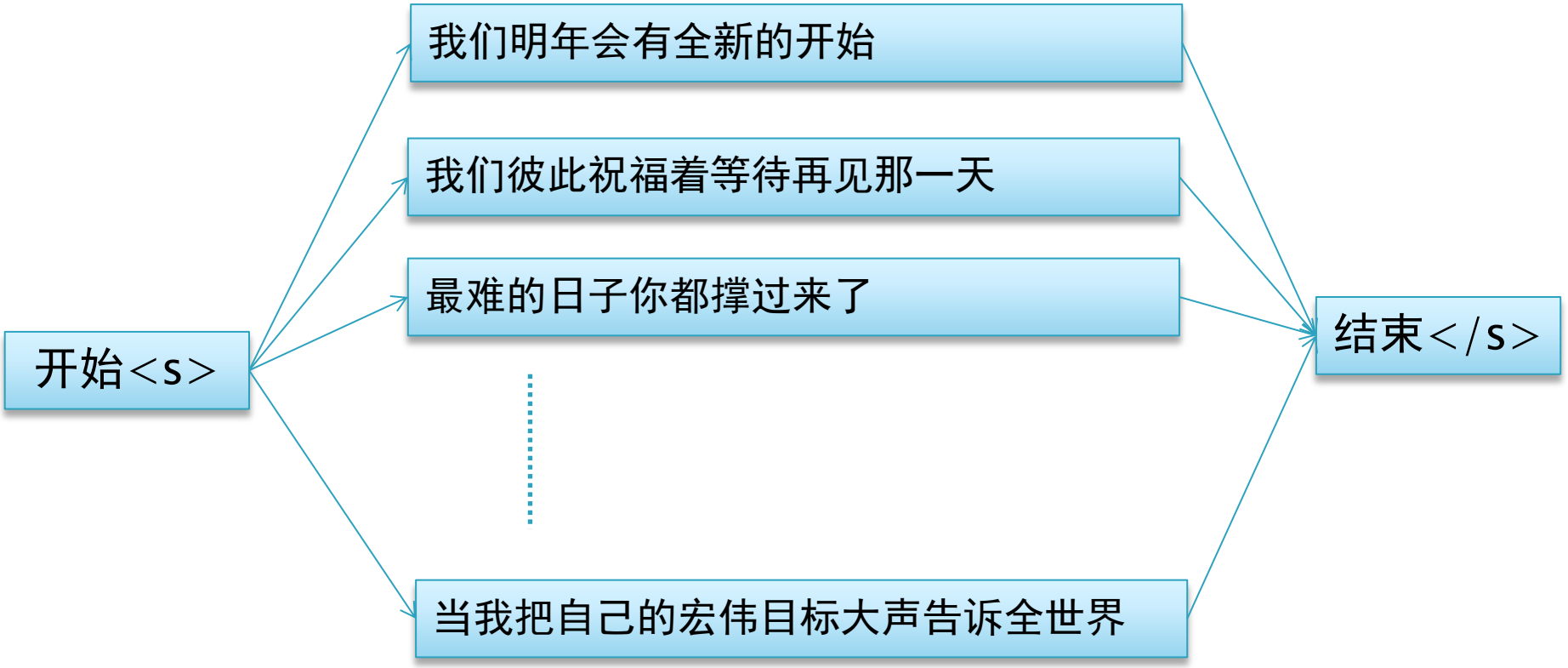
- 要找到最可能的词序列，必须使上式右侧两项的乘积最大。第一项由语言模型决定，第二项由声学模型决定。

语言模型应用

- 语言模型是针对某种语言的概率模型，目的是建立能够描述给定词序列在语言中出现的概率分布。

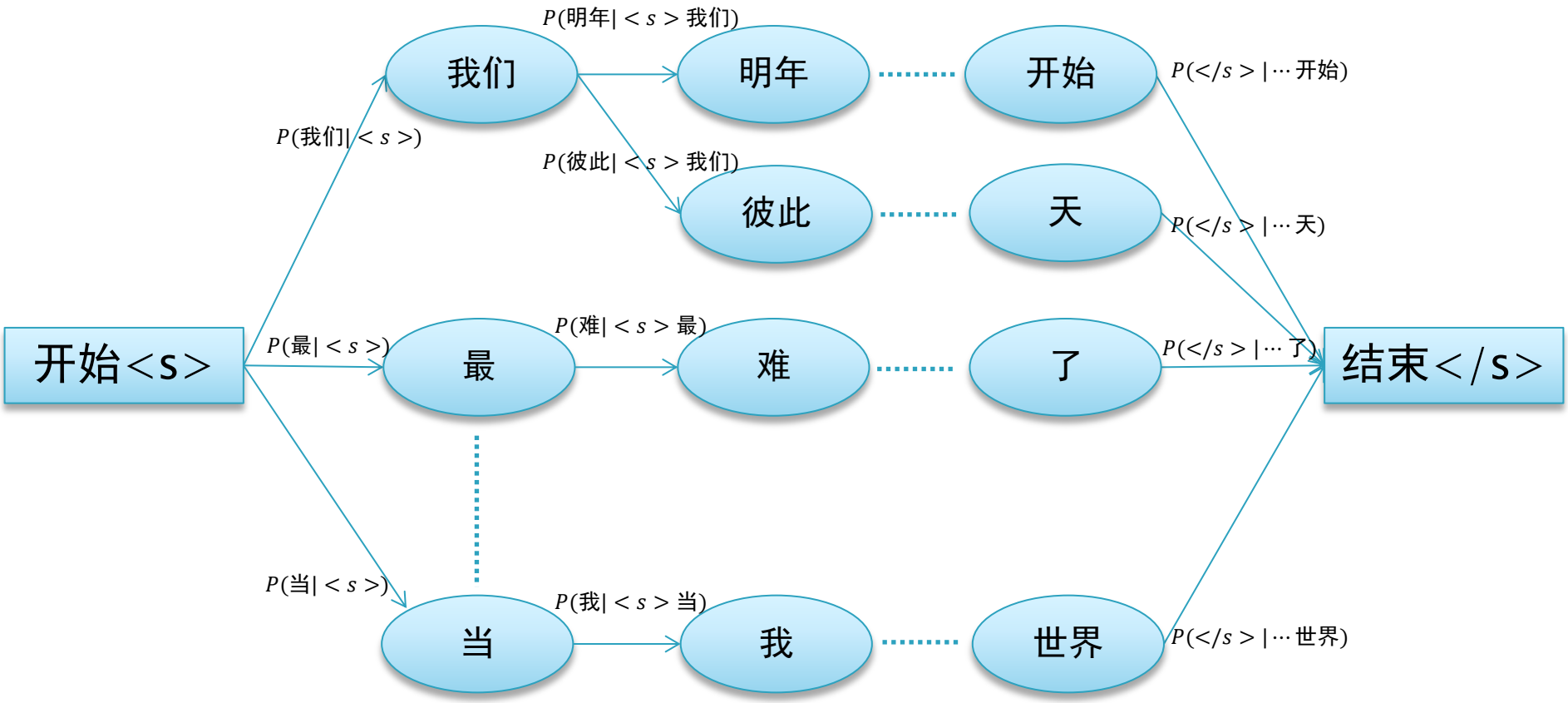
- 语言模型广泛应用于语音识别、机器翻译、输入法等产品中。

不同句子：各种词序列


开始<s>
我们明年会有全新的开始
我们彼此祝福着等待再见那一天
最难的日子你都撑过来了
当我把自己的宏伟目标大声告诉全世界
结束</s>

概率模型

$$P(w_1 \cdots w_{m-1} w_m)$$
$$= P(w_1)P(w_2|w_1)P(w_3|w_1 w_2) \cdots P(w_m|w_{m-n+1} w_{m-n+2} \cdots w_{m-1})$$



语言模型分类

- N-gram：
  - 预测的词概率值依赖于前n-1个词，更长距离上下文依赖被忽略。
  - 效率高，目前仍是主流！

- RNN/LSTM：
  - 将每个词映射到一个紧凑的连续向量空间，该空间使用相对小的参数集合，并使用循环连接来建模长距离上下文依赖。
  - 对计算量要求苛刻，处理大量数据时缓慢的训练速度限制了RNN/LSTM的使用。

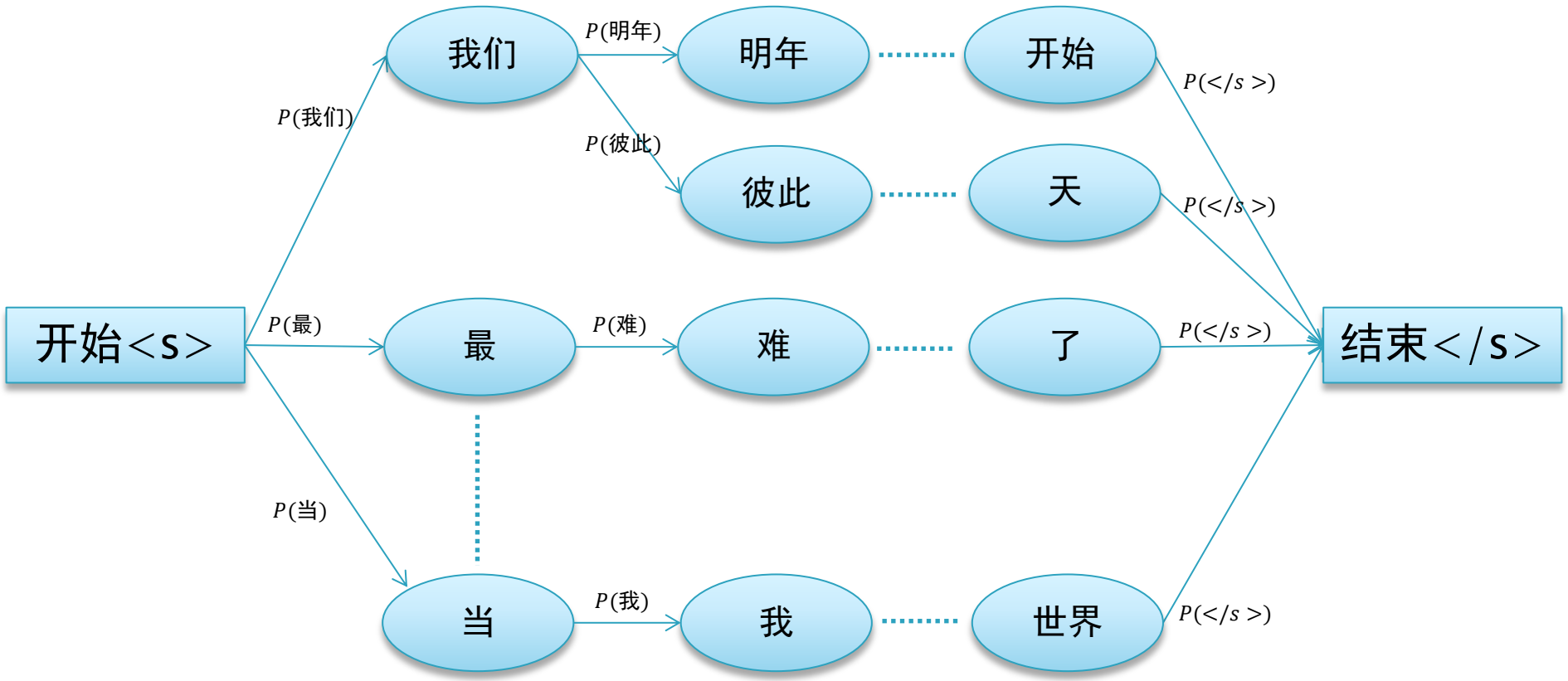
N-gram语言模型

- 1980年由Fred Jelinek和他的同事提出N-gram统计语言模型。每个预测变量w_m只与长度为$n-1$的上下文有关：

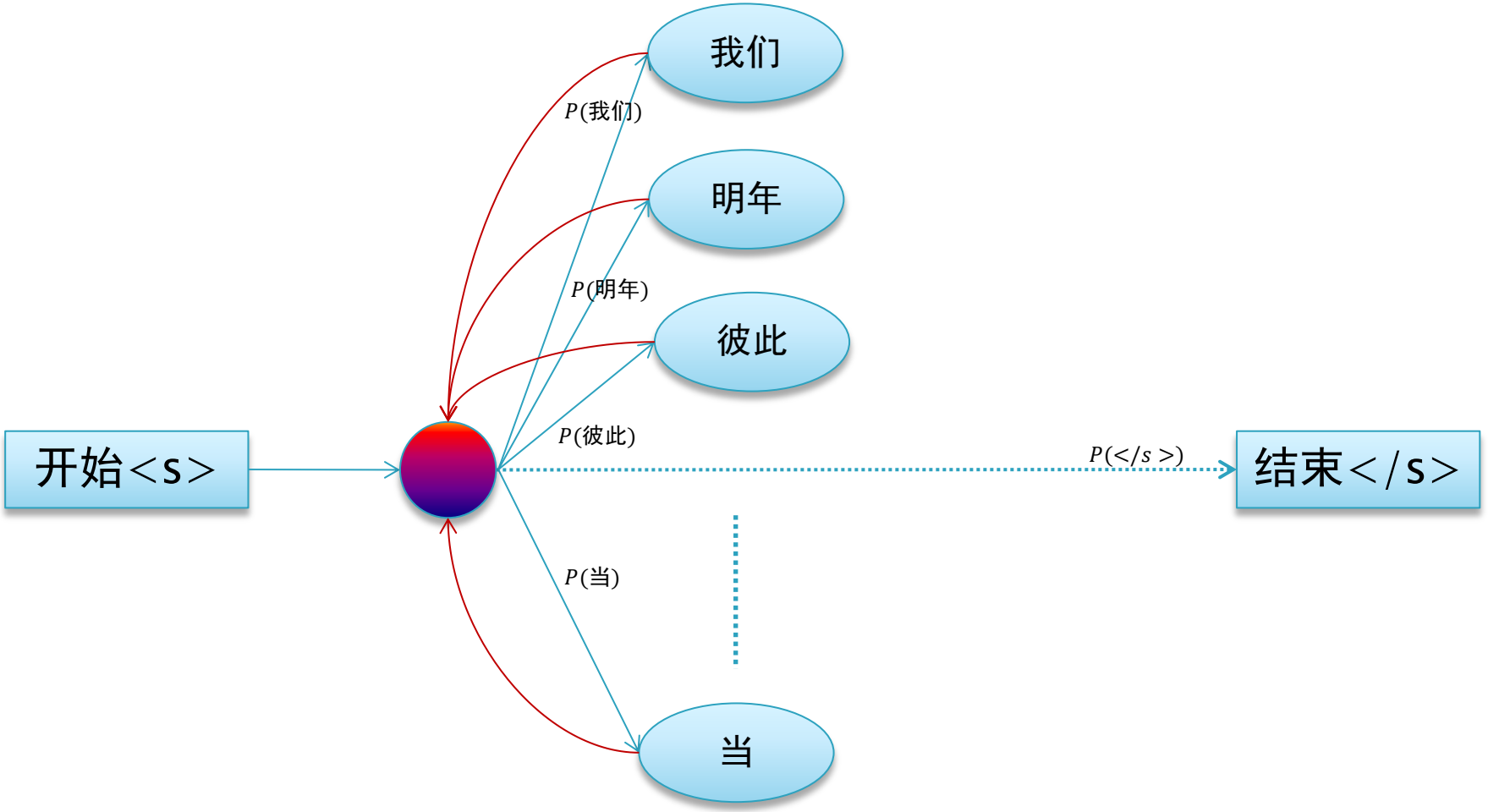
$$P(w_m|w_1 \cdots w_{m-1}) = P(w_m|w_{m-n+1}w_{m-n+2} \cdots w_{m-1})$$

- 实际应用中$1 \leq n \leq 7$，$n = 1,2,3$时，相应的模型分别成为Unigram, Bigram, Trigram语言模型。

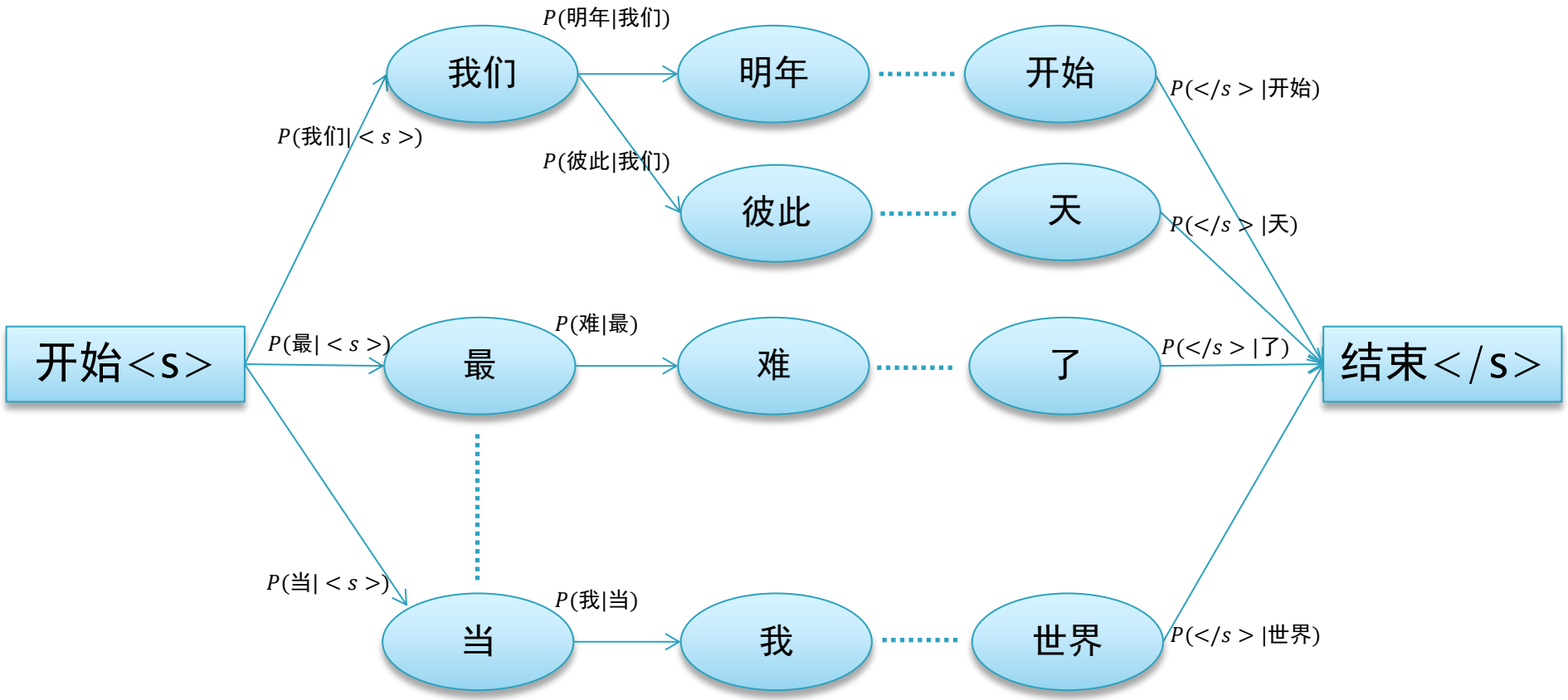
N-gram语言模型：Unigram



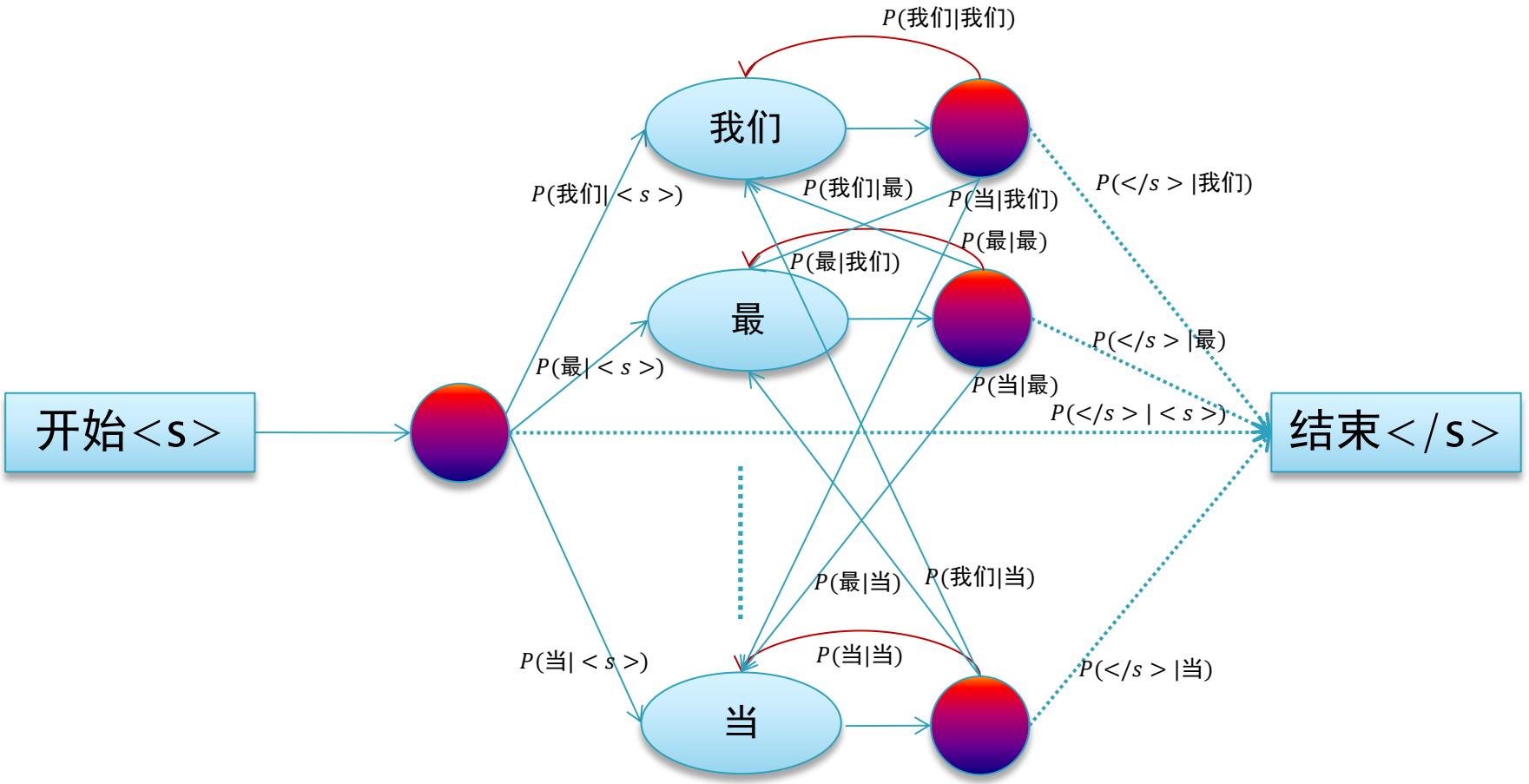
N-gram语言模型：Unigram



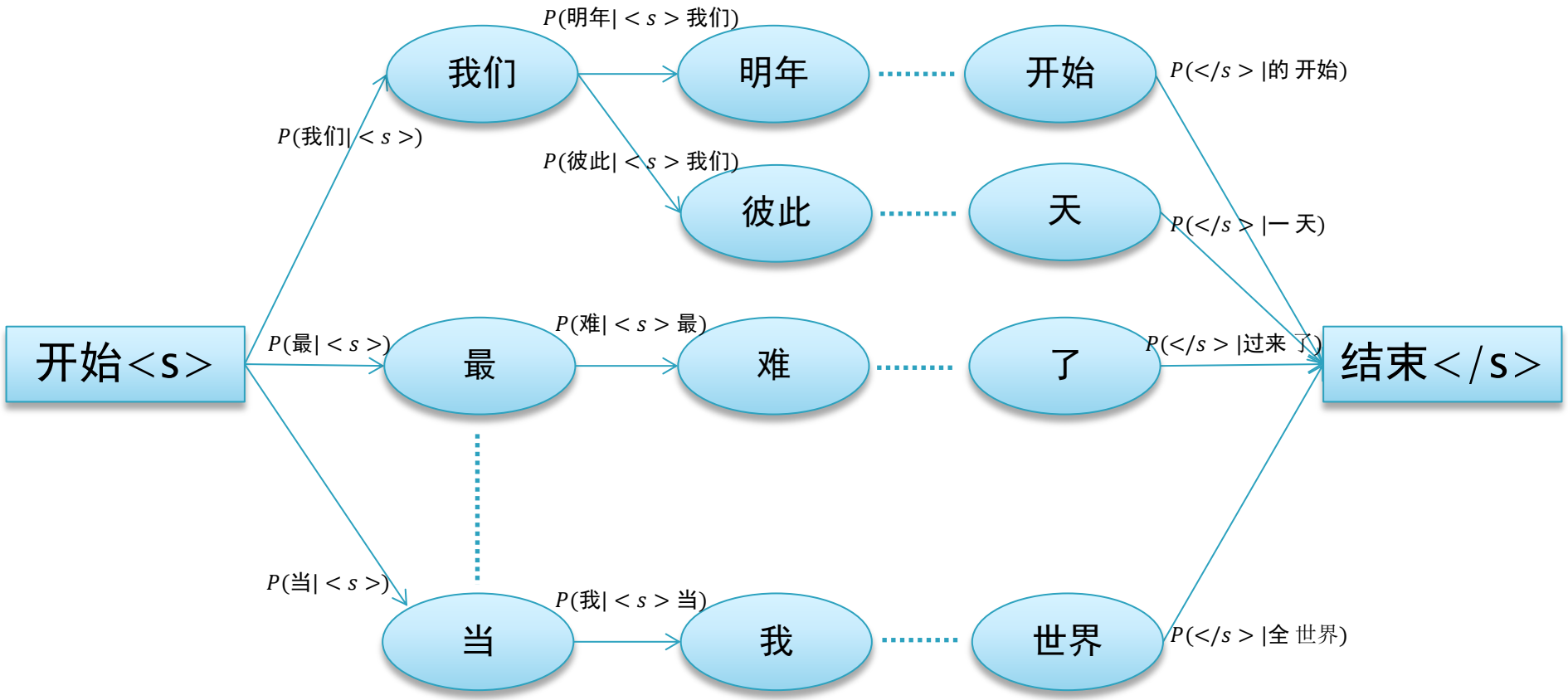
N-gram语言模型：Bigram



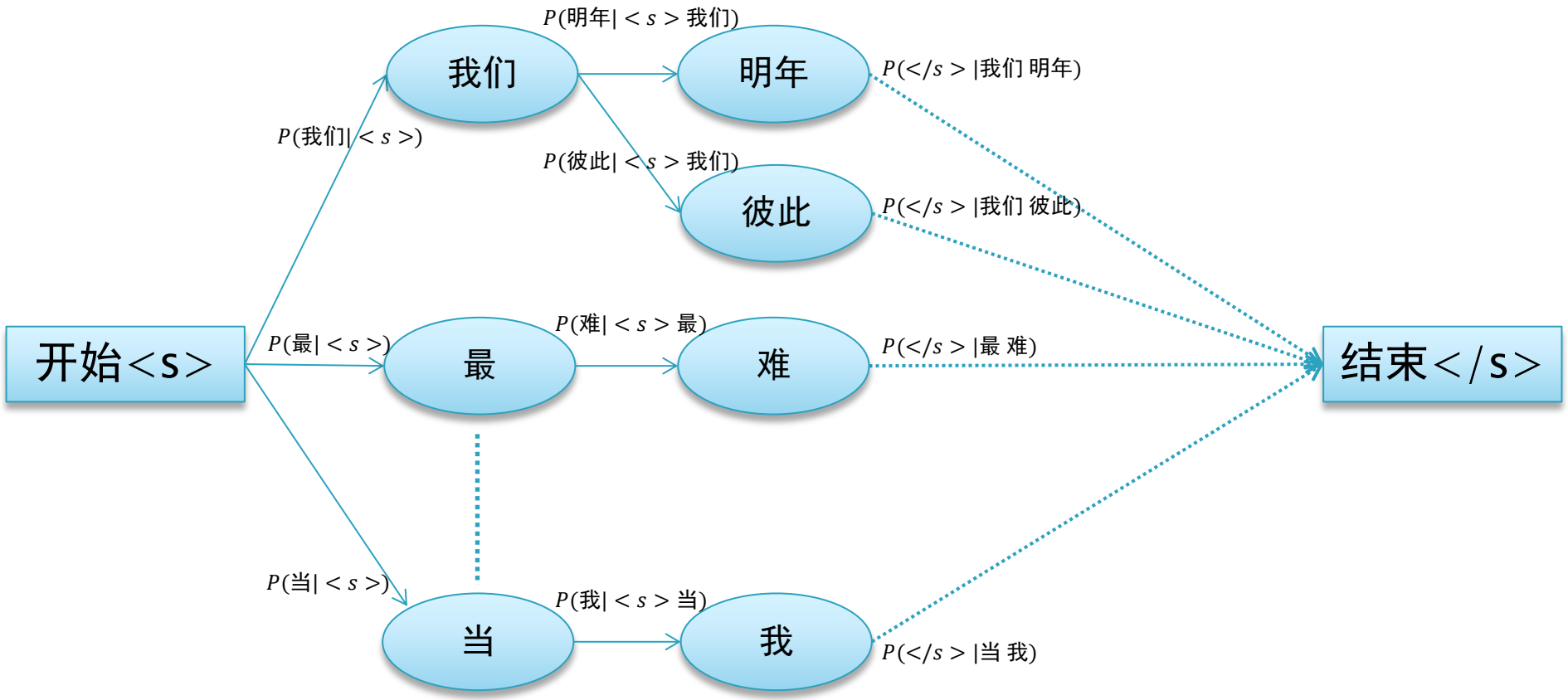
N-gram语言模型：Bigram



N-gram语言模型：Trigram



N-gram语言模型：Trigram



N-gram的训练

- N-gram 概率从文本语料估计得到，可简单地计算词的出现个数。

- 文本语料：

  “我们明年会有全新的开始”

  “我们彼此祝福着等待再见那一天”

  “最难的日子你都撑过来了”

  .....

  “当我把自己的宏伟目标大声告诉全世界”

N-gram的训练：Unigram

- 假设有1000句语料，总共有20000个词
  - “我们”出现100次，“明天”出现30次，“日子”出现10次，……
  - 总共有21000个词标签，其中包括1000句的结束符</s>。

- Unigram计算：
  - P(“我们”) = 100/21000
  - P(“明天”) = 30/21000
  - P(“日子”) = 10/21000
  - P(</s>) = 1000/21000

N-gram的训练：Bigram

- 假设1000句语料，出现：
  - 10句以“我们”开头，5句以“明天”开头，……
  - 2句以“日子”结尾，……
  - 1次出现“我们明年”，3次出现“我们彼此”，……

- 则Bigram计算如下：
  - P(“我们”|<s>) = 10/1000
  - P(“明天”|<s>) = 5/1000
  - P(“日子”|</s>) = 2/1000
  - P(“明年”|“我们”) = 1/100 ---“我们”出现100次
  - P(“彼此”|“我们”) = 3/100

N-gram的训练：Trigram

- 概率计算

$$P(w_3|w_1w_2) = count(w_1w_2w_3)/count(w_1w_2)$$

- 假设“我们明天”出现2次，“我们明天开始”出现1次，则

P(“开始”|“我们明天”) = 1/2

评价指标—Perplexity

- 给定句子S，包含词序列$w_1 \cdots w_N$，N是句子长度，则Perplexity表示为：

$$PP(S) = 2^{-\frac{1}{N}\sum_i log(P(w_i))}$$

- Perplexity又称困惑度(PPL)， PPL越小，$P(w_i)$则越大，句子S出现的概率就越高。

- 理论上，困惑度越小，语言模型越好，预测能力越强。实际应用，也要考虑领域匹配！

平滑技术

- 统计语料有限，存在数据稀疏，导致零概率问题。

- 平滑技术
  - 折扣法(Discounting)：从已有观察概率调配一点给未观察概率。
  - 插值法(Interpolation)：将高阶模型和低阶模型做线性组合。
  - 回退法(Back-off)：基于低阶模型估计未观察到的高阶模型。

平滑技术—Good-Turing(古德-图灵)折扣法

- 算法：
  - 设总词数为N，折扣前出现1次的词数为N_1，出现c次的词数为N_c。
  - 折扣后，概率P^*(出现0次的词) $= \frac{N_1}{N}$，出现次数$c^* = \frac{(c+1)N_{c+1}}{N_c}$。
- 例子：
  - 分词后句子语料（假设只有2句）：
    - “我们 明年 会 有 全新 的 开始”
    - “我们 彼此 祝福 着 等待 再见 那 一 天”
  - 词频数：“我们”2次，“明年”1次，……，“天”1次
  - 折扣前：$N = 16, N_1 = 15, N_2 = 1$
  - 折扣后：${N_0}^* = \frac{N_1}{N} = \frac{15}{16}, {N_1}^* = \frac{(1+1)N_2}{N_1} = \frac{2}{15}$

平滑技术—插值法

- 为了避免出现$P(w)=0$或接近于零的情况，可以用三元、二元和一元相对概率做插值 。

$$\hat{P}(w_n|w_{n-2}w_{n-1}) = \lambda_1 P(w_n|w_{n-2}w_{n-1}) + \lambda_2 P(w_n|w_{n-1}) + \lambda_3 P(w_n)$$

其中$\lambda_1+\lambda_2+\lambda_3=1$。

平滑技术—回退法

- 要计算$P(w_n|w_{n-2}w_{n-1})$，但没有相关的三元统计，可以使用二元语法来估计：

$$P(w_n|w_{n-2}w_{n-1}) = backoff(w_{n-2}w_{n-1})P(w_n|w_{n-1})$$

- 如果没有相关的二元统计，那么我们就用一元模型估计。

语言模型格式(ARPA-MIT)

```
\data\
ngram 1=19979
ngram 2=4987955
ngram 3=6136155

\1-grams:
-1.6682  A       -2.2371
-5.5975  A'S     -0.2818
-2.8755  A.      -1.1409
-4.3297  A.'S    -0.5886
-5.1432  A.S     -0.4862
...

\2-grams:
-3.4627  A  BABY     -0.2884
-4.8091  A  BABY'S   -0.1659
-5.4763  A  BACH     -0.4722
-3.6622  A  BACK     -0.8814
...

\3-grams:
-4.3813  !SENT_START    A       CAMBRIDGE
-4.4782  !SENT_START    A       CAMEL
-4.0196  !SENT_START    A       CAMERA
-4.9004  !SENT_START    A       CAMP
-3.4319  !SENT_START    A       CAMPAIGN
...
\end\
```

\data
ngram 1=nr # 一元语言模型
ngram 2=nr # 二元语言模型
ngram 3=nr # 三元语言模型

\1-grams:
pro_1 word1 back_pro1

\2-grams:
pro_2 word1 word2 back_pro2

\3-grams:
pro_3 word1 word2 word3

\end\

计算出现三个词出现的概率

P(word3|word1,word2)

```
if(存在(word1,word2,word3)的三元模型){
      return pro_3(word1,word2,word3) ;
}else if(存在(word1,word2)二元模型){
      return back_pro2(word1,word2)*P(word3|word2) ;
}else{
      return P(word3 | word2);
}

if(存在(word1,word2)的二元模型){
    return pro_2(word1,word2);
}else{
    return back_pro2(word1)*pro_1(word2) ;
}
```

回退权重计算

- 先采用折扣法计算低阶统计概率，然后计算权重

$$backoff(w_{n-2}w_{n-1}) = \frac{1 - \sum P(w|w_{n-2}w_{n-1})}{\sum P(w'|w_{n-2})}$$

其中w是在训练语料中$w_{n-2}w_{n-1}$之后出现的词，w'是在训练语料中$w_{n-2}w_{n-1}$之后未出现的词。

训练工具

- CMU LM Toolkit (N-gram, 支持Unix平台)
- SRI LM Toolkit (N-gram, 支持Unix/Windows平台)
- CUED-RNNLM Toolkit (可用于HTK/Kaldi)

SRILM

- SRILM是著名的约翰霍普金斯夏季研讨会（Johns Hopkins Summer Workshop）的产物，诞生于1995年，由SRI实验室的Andreas Stolcke负责开发维护。

分词和预处理

- 对于尚未分词的文本进行分词（采用分词工具，如斯坦福大学分词器）

- 处理已经分好词的文本
  - 奇怪的符号，比如α @等
  - 阿拉伯数字
  - 空白行，空格

用SRILM训练语言模型

1、从语料库中生成n-gram计数文件

ngram-count -text *.txt -order 3 -write

参数-text指向输入文件，-order指向生成几元的n-gram,即n,此处为3元，-write指向输出文件

2、从上一步生成的计数文件中训练语言模型

ngram-count -read *.txt.count -order 3 -lm name_of_lm -interpolate -kndiscount

生成语言模型name_of_lm为ARPA文件格式，最后的两个参数是平滑算法，-interpolate为插值平滑，-kndiscount为modified Kneser-Ney打折法，这两个是联合使用的。

3、压缩语言模型

gzip -c name_of_lm > name_of_lm.gz

用SRILM训练语言模型

4、模型融合

#用于多个语言模型之间插值合并，以期望改善模型的效果

ngram -lm ${mainlm} -order 3 -mix-lm ${mixlm} -lambda 0.8 -write-lm ${mergelm}

-mix-lm 用于插值的第二个ngram模型，-lm是第一个ngram模型

-lambda 主模型（-lm对应模型）的插值比例，0~1，默认是0.5

在合并语言模型之前，可以使用脚本计算出最好的比例，参考srilm的compute-best-mix脚本

用SRILM训练语言模型

5、测试语言模型的困惑度(perplexity)

命令行：ngram -lm MinPu-lm8.3gram -order 3 -ppl text.txt -debug 0 > result_ppl_MinPu-lm8.txt

得到以下结果：

file text.txt: 170 sentences, 1249 words, 279 OOVs

0 zeroprobs, logprob= -3778.304 ppl= 2062.06 ppl1= 7855.219

其中ppl表示困惑度，越低越好。

Thank you!

Any questions?