深蓝学院
shenlanxueyuan.com

音频语音与语言处理研究组
SLP
Audio, Speech and Language Processing Group NPU

语音识别：从入门到精通

第七讲：语言模型

主讲人 吕航

西北工业大学
博士
hanglyu1991@gmail.com

内容提要

- 统计语言模型
- N-gram语言模型与评价方法
- 平滑算法(Smoothing)
  - 拉普拉斯平滑(Laplace Smoothing/Add-one Smoothing)
  - 古德图灵平滑(Good-turing Smoothing)
  - 插值(Interpolation) 与 回退(Back off)
  - 卡茨平滑(Katz Smoothing)
  - 克奈瑟-内平滑(Kneser-Ney Smoothing)
- 语言模型的存储格式—APRA Format 及工具包
- RNN语言模型
- 其它语言模型思想简介(Class-based N-gram & Cache Model)
- 大词汇量连续语音识别梳理

统计语言模型

一个统计语言模型包含一个有限集合V，和一个函数$p(x_1, x_2, \dots, x_n)$：

1. 对于任意$\langle x_1, \cdots, x_n \rangle \in v^+$， $p(x_1, x_2, \dots, x_n) \geq 0$

2. $$\sum_{<x_1,\dots,x_n>\in v^+} p(x_1, x_2, \dots, x_n) = 1$$

简言之：统计语言模型是所有词序列上的一个概率分布。

统计语言模型

问：统计语言模型有什么用？

1.它可以给我们任意词序列的概率，即帮助我们确定哪个词序列可能性大。

2.**给定一个词序列，可以预测下一个最可能出现的词语!** [用于ASR，MT等]

给定一个词序列$S = (w_1, \dots, w_n)$，它的概率可以表示为：

$$P(S) = P(x_1 = w_1, \dots x_n = w_n) = P(w_1^n)$$

$$= P(w_1)P(w_2|w_1)P(w_3|w_1^2) \dots P(w_n|w_1^{n-1})$$

$$= \prod_{k=1}^{n} P(w_k|w_1^{k-1})$$

统计语言模型

如何获得语言模型中的概率？ 在语料库(Corpus)中计数。

训练集(training-set)与测试集(test-set)
保留集(held-out set): 从训练集中分离，用来计算一些其它参数，
如插值模型中的插值系数。

语言模型中一些常见术语：

1. I uh gave a re- report yesterday
2. 我 是 不 是 老 了

有声停顿(fillers/filled pauses): 如uh就是一个没有实际意义的有声停顿。
截断(fragment):表示没有说完整，如re-。
词目(lemma)：词语主干(stem)相同，比如dogs和dog是一个词目。
词形(wordforms)：完整的词语样子，比如dogs和dog是两个词形。
型(type):语料库或者字典中不同单词的数目。
例(token):语料中单词数目。(数数)
字典(vocabulary)：语言模型的基本组件，规定了我对那些元素进行统计。

例句：It is too expensive to buy

$$P(buy \mid It\ is\ too\ expensive\ to) = \frac{C(It\ is\ too\ expensive\ to\ buy)}{C(It\ is\ too\ expensive\ to)}$$

问题：当历史信息越长，我们越难在语料库中发现完全一致的序列。

直觉(Intuition)：用最近的几个历史词代替整个历史词串，从而近似。

回忆Markov assumption

N-gram：用前N-1个词作为历史，估计当前(第N个)词。

$$P(w_i|w_1^{i-1}) = P(w_i|w_{i-N+1}^{i-1})$$

$$bigram\ example: P(w_i|w_1^{i-1}) = P(w_i|w_{i-1})$$

$$P(S) = P(x_1 = w_1, \dots x_n = w_n) = P(w_1^n)$$

$$= P(w_1)P(w_2|w_1)P(w_3|w_1^2) \dots P(w_n|w_1^{n-1})$$

$$= \prod_{k=1}^{n} P(w_k|w_1^{k-1}) \Rightarrow = \prod_{k=1}^{n} P(w_k|w_{k-N+1}^{k-1})$$

N-gram
语言模型

N-gram语言模型与评价方法

如何估计N-gram？　最大似然方法(MLE)→数数

$$P\left(w_i \middle| w_{i-N+1}^{i-1}\right) = \frac{C\left(w_{i-N+1}^{i-1} w_i\right)}{C\left(w_{i-N+1}^{i-1}\right)}$$

$$bigram\ example:\ P(w_i|w_{i-1}) = \frac{C(w_{i-1}w_i)}{C(w_{i-1})}$$

开头结尾如何处理，与未知词？

习惯上，ASR领域用<s>和</s>来标记开头结尾，并方便统计。
没有在vocabulary中的词(OOV, out of vocabulary)，一般标记为<UNK>。

如何评估？ 1. 根据应用实地测试 2. 困惑度(Perplexity)

在测试集$W = w_1, w_2, \ldots, w_N$, **困惑度**就是用单词数归一化后的测试集概率：

$$PP(W) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}}$$

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

$$= \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_{i-N+1}^{i-1})}}$$

小贴士：
在工具包里，
你会见到PPL和PPL1

PPL：考虑词数和句子数
(i.e. 考虑</s>)

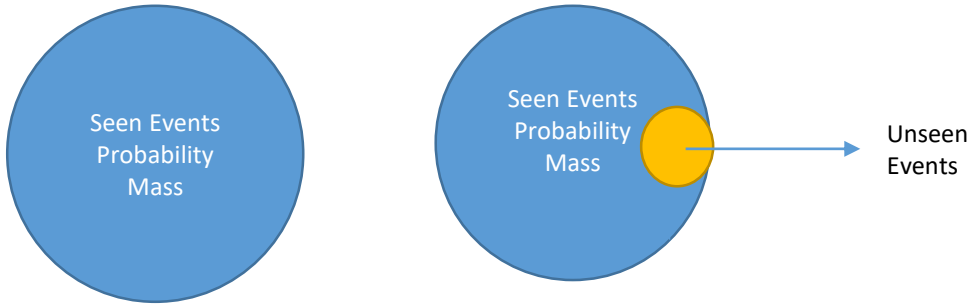
PPL1：只考虑词数

平滑算法(Smoothing)

由于语料的稀疏性(sparse data), 有些词序列找不到怎么办？

平滑！

将一部分看见的事件概率量分给未看见的事件。



平滑算法(Smoothing)—Laplace Smoothing

拉普拉斯平滑(Laplace Smoothing/Add-one Smoothing)

Intuition：将每个计数加一，从而使得任何词序列都有计数。

以$unigram$为例：$P(w_i)=\frac{C(w_i)}{N}$，N为总$token$数。$P_{laplace}(w_i)=\frac{C(w_i)+1}{N+V}$

那么$bigram$呢？ $P_{laplace}(w_i|w_{i-1})=\frac{C(w_{i-1}w_i)+1}{C(w_{i-1})+V}$

平滑算法(Smoothing)—Laplace Smoothing

以$unigram$为例：$P(w_i) = \frac{C(w_i)}{N}$，$N$为总$token$数。$P_{laplace}(w_i) = \frac{C(w_i)+1}{N+V}$

两个重要的概念：　(我们将词w_i的$token$数简记为c_i)

1. 调整计数(adjusted count) $--c^*$：描述平滑算法仅对分子的影响。

$$Laplace\ smoothing\text{为例}c_i^* = (c_i + 1)\frac{N}{N+V}$$

2. 相对打折 (relative discount)/打 折率(discount ratio) $--d_c$: 打折计数和原计数的比率。

$$d_c = \frac{c^*}{c}$$

Laplace Smoothing缺点：　原来计数量较高的词序列，概率削减严重。

--> Add-delta smoothing (缓解)

平滑算法(Smoothing)—Good-turing Smoothing

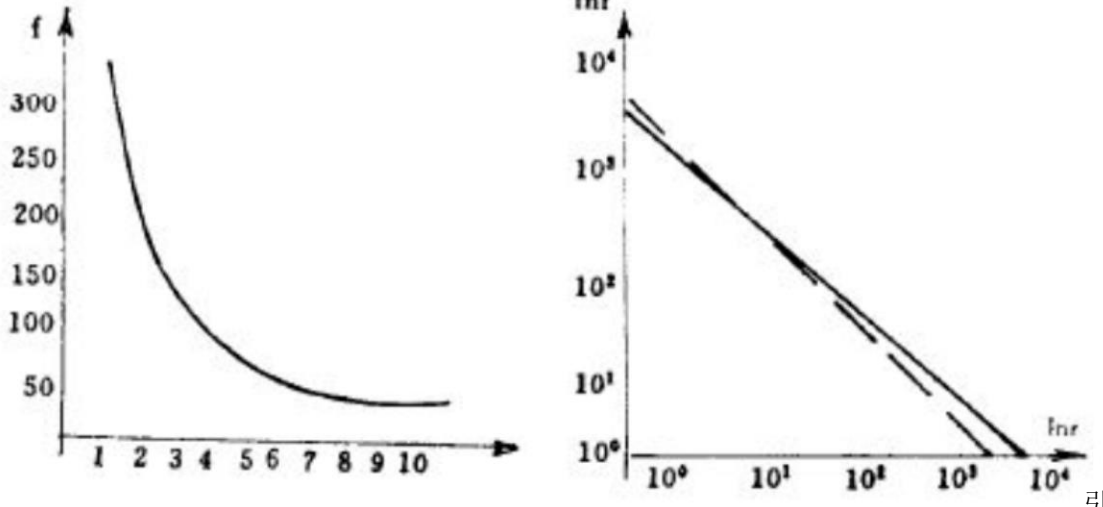
齐夫(Zipf)定律:

Given a large sample of words used, the frequency of any word is inversely proportional to its rank in the frequency table.—Wikipedia

自然语言语料库中，一个词出现的频率与它在频率表里面的排名成反比。

告诉了我们什么?

语言中大部分词都是低频词，只有很少的常用词。



引自网络

平滑算法(Smoothing)—Good-turing Smoothing

古德图灵平滑(Good-turing Smoothing)

Intuition：用你看见过一次的事情(Seen Once)估计你未看见的事件(Unseen Events)，并依次类推。

频率c出现的频数(frequency of frequency c) − −Nc

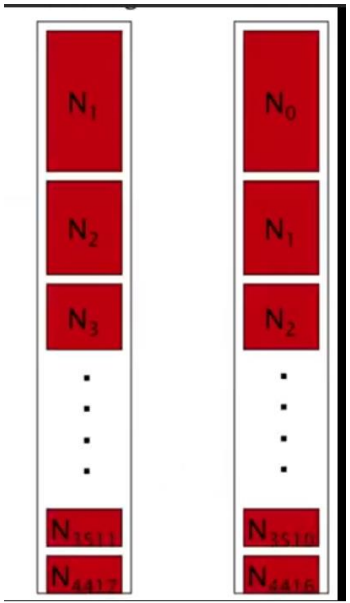
$$N_c = \sum_{x:count(x)=c} 1$$

算法：

$$c^* = （c+1）\frac{N_{c+1}}{N_c}$$

思考一下如何由上式获得

$$P^*_{GT}(\text{Unseen Events}) = \frac{N_1}{N}$$ 也称为遗漏量(missing mass)



引自Dan Jurafsky，Stanford University，NLP

平滑算法(Smoothing)—Good-turing Smoothing

问题：显然，如果一个词在语料中出现了5000次的词序列(e.g. the)，但是没有出现5001次的词序列。难道这个概率变为0？

方法1: 当$c > \mathrm{k}\,(\mathrm{e.g.}\,\mathrm{k} = 5)$时，$N_c \approx \alpha c^{\beta}$，其中$\alpha$和$\beta$为参数，对$N_c$进行平滑。

方法2：认为$\mathrm{c} > \mathrm{k}\,(\mathrm{katz建议k} = 5)$，认为计数可靠不进行打折。 [$Katz$ 1987]

$$\begin{cases} c^* = c & c > k \\ c^* = \dfrac{(c+1)\dfrac{N_{c+1}}{N_c} - c\dfrac{(k+1)N_{k+1}}{N_1}}{1 - \dfrac{(k+1)N_{k+1}}{N_1}} & 1 \le c \le k \end{cases}$$

Good-turing算法很少单独使用，通常结合后面的算法使用

平滑算法(Smoothing)—插值(Interpolation) 与 回退(Back off)

插值(Interpolation)法

Intuition：从所有**N-grams**估计中，把所有的概率估计混合。例如，我们优化一个tri-gram模型，我们将统计的tri-gram，bigram和unigram计数进行插值。

回退(Back off)法

Intuition：如果有非零的高阶语言模型，我们直接只用。只有当高级语言模型存在计数零时，我们回退到低阶语言模型。(递归)

例如，一个tri-gram，某些概率为0，我们将tri-gram非零计数打折后，获得遗漏量，将遗漏量用bi-gram进行重分配，依次类推。

平滑算法(Smoothing)—插值(Interpolation) 与 回退(Back off)

以tri-gram为例介绍插值法：

$$P_{ML}(w_i|w_{i-2}w_{i-1}) = \frac{C(w_{i-2}w_{i-1}w_i)}{C(w_{i-2}w_{i-1})}$$

$$P_{ML}(w_i|w_{i-1}) = \frac{C(w_{i-1}w_i)}{C(w_{i-1})}$$

$$P_{ML}(w_i) = \frac{C(w_i)}{N}$$

$$P_{interp}(w_i|w_{i-2}w_{i-1}) = \lambda_1 P_{ML}(w_i|w_{i-2}w_{i-1}) + \lambda_2 P_{ML}(w_i|w_{i-1}) + \lambda_3 P_{ML}(w_i)$$

$$\lambda_1 + \lambda_2 + \lambda_3 = 1$$

小贴士：
参数$\lambda_1, \lambda_2, \lambda_3$可以是经验值，也可以由MLE等优化算法获得。

平滑算法(Smoothing)—插值(Interpolation) 与 回退(Back off)

如何确定系数？ 在held-out set上使用MLE的方法

目标： $argmax_{\lambda_1,\lambda_2,\lambda_3} L(\lambda_1,\lambda_2,\lambda_3)$

$$L(\lambda_1,\lambda_2,\lambda_3) = \sum_{w_i,w_{i-1},w_{i-2}} c'(w_{i-2},w_{i-1},w_i) \log[\lambda_1 P_{ML}(w_i|w_{i-2}w_{i-1}) + \lambda_2 P_{ML}(w_i|w_{i-1}) + \lambda_3 P_{ML}(w_i)]$$

从Held-out集合上统计

平滑算法(Smoothing)—插值(Interpolation) 与 回退(Back off)

Bucketing方法扩展Interpolation：

在实际工程中，参数$\lambda_1, \lambda_2, \lambda_3$通常不是全局的，而是根据历史(i.e. $P_{ML}(w_i|w_{i-2}w_{i-1})$中的$w_{i-2}w_{i-1}$)计数做一些调整。

$$\lambda_1, \lambda_2, \lambda_3 \rightarrow \lambda_1^{(k)}, \lambda_2^{(k)}, \lambda_3^{(k)}$$

根据历史计数$c(w_{i-2}w_{i-1})$定义函数 $(k) = \pi(w_{i-2}, w_{i-1})$:　　[简例如下]

$$\Pi(w_{i-2}, w_{i-1}) = 1 \quad if\ c(w_{i-2}, w_{i-1}) > 0$$

$$\Pi(w_{i-2}, w_{i-1}) = 2 \quad if\ c(w_{i-2}, w_{i-1}) = 0\ and\ c(w_{i-1}) > 0$$

$$\Pi(w_{i-2}, w_{i-1}) = 3 \quad otherwise$$

$$P_{interp}(w_i|w_{i-2}w_{i-1}) = \lambda_1^{(k)} P_{ML}(w_i|w_{i-2}w_{i-1}) + \lambda_1^{(k)} P_{ML}(w_i|w_{i-1}) + \lambda_1^{(k)} P_{ML}(w_i)$$

平滑算法(Smoothing)—Katz Smoothing

卡茨平滑(Katz Smoothing)—递归回退算法

Intuition：若N阶语言模型存在，直接使用打折后的概率(常使用Good-turing算法进行打折)；若高阶语言模型不存在(i.e. unseen events)，将打折节省出的概率量，依照N-1阶的语言模型概率进行分配，依此类推。

$$P_{katz}\left(w_i \middle| w_{i-N+1}^{i-1}\right) = \begin{cases} P^*\left(w_i \middle| w_{i-N+1}^{i-1}\right) & C\left(w_{i-N+1}^{i}\right) > 0 \\ \alpha\left(w_{i-N+1}^{i-1}\right) P_{katz}\left(w_i \middle| w_{i-N+2}^{i-1}\right) & \text{其它} \end{cases}$$

$\alpha\left(w_{i-N+1}^{i-1}\right)$为归一化系数

平滑算法(Smoothing)—Katz Smoothing

卡茨平滑(Katz Smoothing)—归一化系数怎么求?

思想：将打折获得的概率量分配给未知事件。

$$\alpha\left(w_{i-N+1}^{i-1}\right)=\frac{1-\sum_{w_i:c\left(w_{i-N+1}^{i}\right)>0}P^*\left(w_i\middle|w_{i-N+1}^{i-1}\right)}{\sum_{w_i:c\left(w_{i-N+1}^{i}\right)=0}\mathrm{P}_{katz}\left(w_i\middle|w_{i-N+2}^{i-1}\right)}$$

分子：1－对N阶语言模型存在的序列打折 = 打折获得的概率量。
分母：N阶不存在的部分，用N-1阶按比例分配

$$=\frac{1-\sum_{w_i:c\left(w_{i-N+1}^{i}\right)>0}P^*\left(w_i\middle|w_{i-N+1}^{i-1}\right)}{1-\sum_{w_i:c\left(w_{i-N+1}^{i}\right)>0}P^*\left(w_i\middle|w_{i-N+2}^{i-1}\right)}$$

分母为什么可以这么变？

N阶存在，那么N-1阶必存在，所以对这类词序列以久按照打折算法打折。

平滑算法(Smoothing)—Kneser-Ney Smoothing

克奈瑟-内平滑(Kneser-Ney Smoothing) [以bigram为例]

一、绝对折扣(absolute discounting)

引自Dan Jurafsky，Stanford University，NLP

c(MLE)	0	1	2	3	4	5	6	7	8	9
c^*(GT)	0.0000270	0.446	1.26	2.24	3.24	4.22	5.19	6.21	7.24	8.25

研究人员对“AP新闻专线”语料的bigram进行Good-turing打折后获得此表

现象：除了0和1以外，其它近似减0.75就可以了。

$$P_{absolute}(w_i|w_{i-1}) = \begin{cases} \dfrac{C(w_{i-1}w_i) - D}{C(w_{i-1})} & C(w_{i-1}w_i) > 0 \\ \alpha(w_i)P(w_i) & otherwise \end{cases}$$

小贴士：实际应用，0和1会特殊处理。如用专有的D值等。

平滑算法(Smoothing)—Kneser-Ney Smoothing

克奈瑟-内平滑(Kneser-Ney Smoothing) [以bigram为例]

现象：我们训练好了一个bigram的语言模型，之后我们准备填空：I'm finding my new ___。我们发现这里填glasses似乎比Zealand更好。但是我语料库里New Zealand很多，导致bigram认为这里填Zealand更好。

Intuition：对于一个词，如果它在语料库中出现更多种不同上下文(context)时，它可能应该有更高的概率。

为了刻画这种想法，我们定义接续概率(continuation probability):

$$P_{continuation}(w_i) = \frac{|\{w_{i-1}: C(w_{i-1}w_i) > 0\}|}{\sum_{w_i} |\{w_{i-1}: C(w_{i-1}w_i) > 0|}$$

平滑算法(Smoothing)—Kneser-Ney Smoothing

克奈瑟-内平滑(Kneser-Ney Smoothing)　[以bigram为例]

$$P_{KN}(w_i|w_{i-1}) = \begin{cases} \dfrac{C(w_{i-1}w_i) - D}{C(w_{i-1})} & C(w_{i-1}w_i) > 0 \\ \alpha(w_i)P_{continuation}(w_i) & otherwise \end{cases}$$

回退式

$$P_{KN}(w_i|w_{i-1}) = \frac{C(w_{i-1}w_i) - D}{C(w_{i-1})} + \beta(w_i)P_{continuation}(w_i)$$

$$= \frac{C(w_{i-1}w_i) - D}{C(w_{i-1})} + \beta(w_i)\frac{|\{w_{i-1}: C(w_{i-1}w_i) > 0\}|}{\sum_{w_i}|\{w_{i-1}: C(w_{i-1}w_i) > 0|}$$

插值式

小贴士：研究表明插值式效果更好。

语言模型的存储格式—APRA Format 及工具包

ARPA Format是N-gram的标准储存模式：是一个ASCII文件，在小标题后跟着一个表，列举出所有非零的N元语法概率。

每个N元语法条目中，依次为：

折扣后对数概率(log10格式存储），
词序列，
回退权重(log10格式存储）

e.g. $\log_{10} P^*(w_i|w_{i-1})$ $w_{i-1}w_i$ $\log \alpha(w_{i-1}w_i)$

```
\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\
```

思考题
1.最高阶语法和</s>结尾的任意阶语法没有回退权重。
2. 插值模型和回退模型都可以如此储存。

引自asimihsan，NLP

语言模型的存储格式—APRA Format 及工具包

工具包：

--SRILM(最常用):

http://www.speech.sri.com/projects/srilm

--KenLM:

https://github.com/kpu/**kenlm**

--KaldiLM:

http://www.danielpovey.com/files/kaldi/kaldi_lm.tar.gz

--IRSTLM:

https://github.com/irstlm-team/

回头看统计语言模型的目标：

$$P(S) = P(x_1 = w_1, \dots x_n = w_n) = P(w_1^n)$$

$$= P(w_1)P(w_2|w_1)P(w_3|w_1^2) \dots P(w_n|w_1^{n-1})$$

$$= \prod_{k=1}^{n} P(w_k|w_1^{k-1})$$

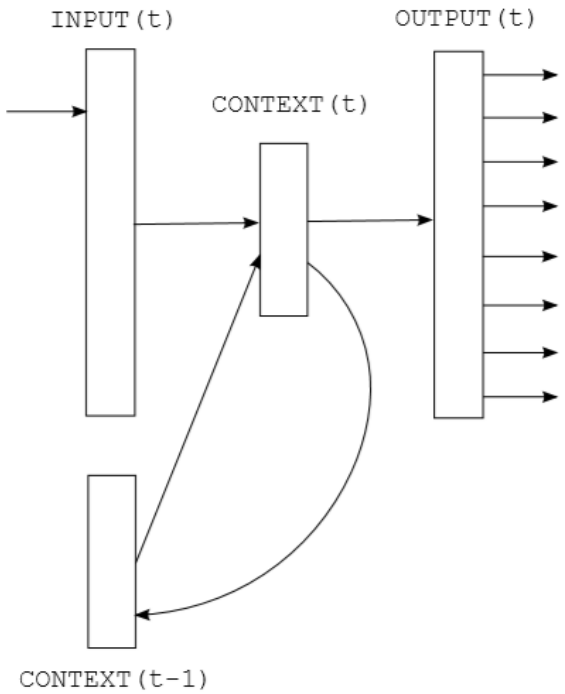
Recurrent Neural Network正好满足这个要求

如果可能它希望依赖所有历史推测当前。

RNN语言模型

Input: vocabulary size one-hot vector
Structure: Simple Recurrent Neural Network
(extend: LSTM, GRU and so on)

Output: vocabulary size vector with softmax
(extend: Top N(e.g. 2k) high frequency words
+ 1 low frequency word bag)



优化：低频词袋法

$$P(w_i(t+1)|w(t),s(t-1)) = \begin{cases} \frac{y_{rare}(t)}{C_{rare}} & \text{if } w_i(t+1) \text{ is rare,} \\ y_i(t) & \text{otherwise} \end{cases}$$

引自Tomas Mikolov，Recurrent neural network based language model

其它语言模型思想简介(Class-based N-gram & Cache Model)

基于类的N元语言模型(Class-based/Clustering N-gram):

Intuition: 当我们进行某种特定有结构任务时(e.g. 订票系统)，我们可以先根据标签将其分类(e.g. 目的地)，后再依据类别信息筛选词(即通过类别信息剔除部分错误)，并可以通过类内词参数共享解决部分unseen问题。

以bigram为例：$P(w_i|w_{i-1}) \approx P(c_i|c_{i-1}) * P(w_i|c_i)$

缓存模型(Cache Model):

Intuition: 如果一个词在句子中用到，那么它很可能被再次用到。

例如：两个人在讨论旅游，它们可能反复用到同一个地名。

大词汇量连续语音识别梳理

我们都学了什么?

语言模型：建模word间的跳转概率。
字典(vocabulary)：提供word到phone的映射，及语言模型建模元素。
HMM：建模phone或triphone等基本单元发声过程。
GMM：建模每个HMM状态的发射概率，即声学似然分。
决策树：triphone等建模单元绑定(共享pdf)，解决数据稀疏问题。
前向后向算法：更新HMM参数。
EM算法：更新GMM参数。
Viterbi算法：解码或对齐。
Embedding Training算法：更新GMM-HMM模型参数，即Viterbi Training。
特征提取：从音频获取MFCC，PLP，Fbank等特征。
DNN：建模每帧观测的后验概率，后转化为似然概率，提供给每个HMM状态。

大词汇量连续语音识别梳理

LM训练

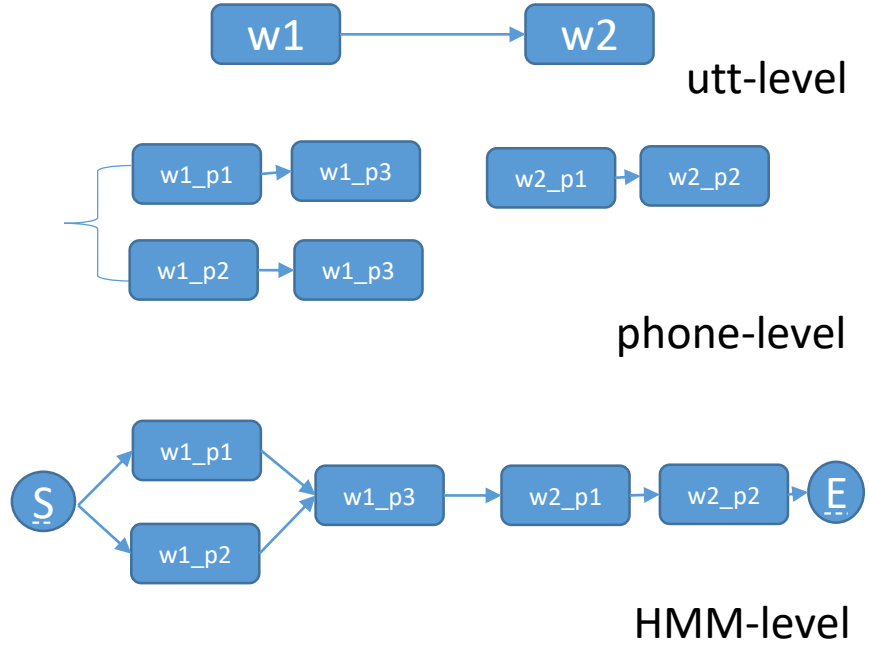
特征提取 + 标注抄本

基于GMM-HMM的Monophone模型训练

基于GMM-HMM的Tri-phone模型训练

获取训练数据对齐

DNN-HMM模型训练



作业

作业1：利用上次作业下载的THCHS30数据，学习使用SRILM。
1.使用SRILM获得经过interpolation式的Kneser-Ney平滑的3-gram以上的语言模型。总结你观察到的一些现象。
2.使用SRILM计算在识别测试集上计算PPL。

作业2：[修改自哥伦比亚大学语音识别课程E6870的一部分]（阅读代码框架）
1.实现N-gram计数 (实验指导书part 2.2)
2.实现Witten-Bell smoothing部分(总结它的Intuition) (实验指导书part 4)

感谢各位聆听!

Thanks for Listening