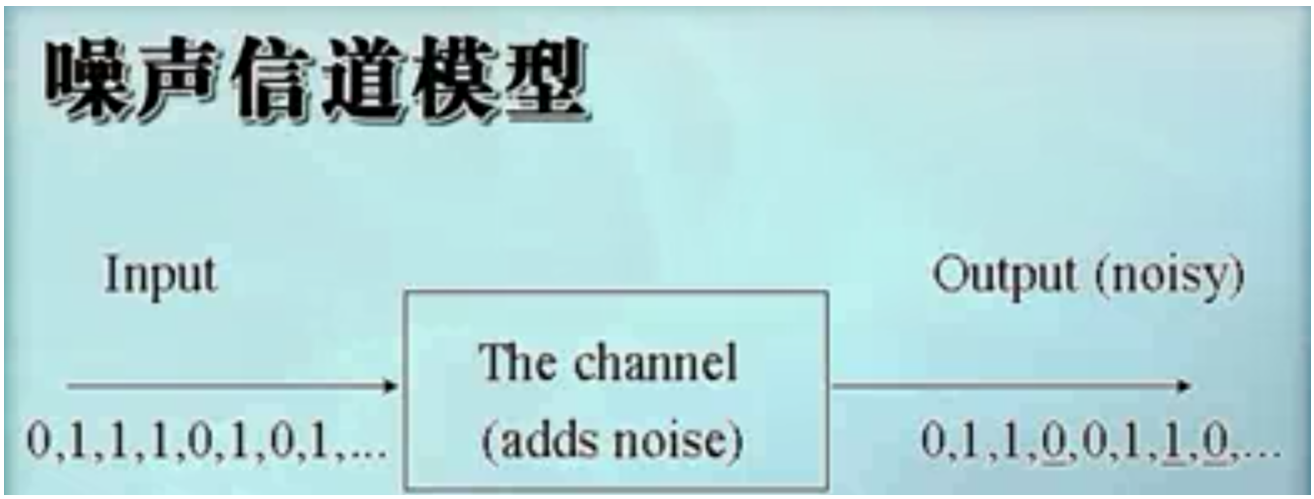
语言模型

为什么需要语言模型

为了说明为什么需要语言模型，首先介绍噪声信道模型

噪声信道模型



根据输出端输出A找到最有可能的输入信号T:

数学表达为：$T = argmax_T(p(T|A))$,

根据贝叶斯定理有$T = argmax_T \frac{p(T)p(A|T)}{p(A)}$ ：

由于T不依赖于A,从而有$T = argmax_T \frac{p(T)p(A|T)}{p(A)} = argmax_T(p(T)p(A|T))$

贝叶斯定理使我们能够交换条件依赖顺序，使得许多任务变得更加简单。比如说对于语音识别任务，相同的声音信号可能对应非常多个词（同音字非常多），但是一个词一般只有一到两个读音，而$P(T)$可以通过大规模语料训练得到，因此问题会变得更加简单。

信源以一定概率发射文本串，channel(信道)对应着将文本串转换为其他信号，事实上NLP中许多任务都可以纳入这个模型之中。

例如，对于语音识别，输入为文本串，输出信号为语音信号（人们真正想表达的文本意义，只是通过发声系统（对应channel），转换为语音信号（对于输出端），语音识别的目的就是根据语音信号，恢复人们想表达的文本信息）；

音字转换，输入为文本串， 输出为拼音串，通过拼音串恢复文本串

机器翻译，输入为目标文本，输出为源文本，通过源语言文本恢复目标语言文本

词性标注的噪声信道解释：

信源：以某种概率发射词性标注序列

噪声信道：根据词性标注序列转化为对应的语句

词性标注目的是通过输出端语句恢复词性标注序列

（事实上，这可能也是人类表达语言的方式）

。。。

上述四个应用中信源都是以一定概率发射的文本串$p(T)$,四个应用只是在噪声信道的定义各不相同，但各个不同应用都需要构造$p(T)$，语言模型应用非常广泛，事实上**越了解语言模型，越能明白许多语言理解问题以及人工智能相关问题都隐藏在这个问题之后。** 语言模型所要求的数据不需要标记，有大量文本数据可以用来进行训练语言模型。

语言模型

语言模型定义

计算一个文本序列$\omega = (w_1, w_2, w_3, \dots w_N)$出现的概率，即：$p(w_1, w_2 \dots w_N)$

$p(w_1, w_2 \dots w_N) = p(w_1)p(w_2|w_1)p(w_3|w_1, w_2) \dots p(w_N|w_1, w_2 \dots w_{N-1})$

概率语言模型面临的主要问题 ：

从上式可以看出概率语言模型的两个主要问题：

1、**参数空间过大**（词表很大，长句子组合概率组合爆炸，假设10000个词，10个字长的参数空间为10000^{10} ）

2、**数据稀疏** ：（根据zipf law ，不管语料库多大，绝大多数序列出现次数很小或根本不出现，概率连乘会使得大部分句子的概率都为零）

概率语言模型的评价方法

基本思想 ：好的语言模型就是给真实的语言序列更高的概率

1、外部评价 ：根据实际应用的效果进行评价（缺点：实际应用最终效果影响很多，很难判断语言模型的好坏）

2、内部评价 ： 交叉熵 和 迷惑度

下面说明为什么交叉熵可以作为语言模型的评价标准：

熵(ertropy 自信息) ：描述一个随机变量的不确定性的度量 $H(x) = -\sum_x p(x)log(p(x))$

熵率 ：$ER = \frac{1}{n}H(x_1^n) = -\frac{1}{n}\sum_{x_1^n} p(x_1^n)logp(x_1^n)$

把语言视为一个平稳的可遍历的随机过程，根据SMB定律，那么：

$H(L) = lim_{n\to \inf} -\frac{1}{n}log(p(x_l^n))$

相对熵(relative entropy) :又称KL距离，

KL(Kullback-Leibler)距离:衡量两个概率分布之间差异的量度（不满足三角不等式和交换律）

$D(p||q) = \sum_{x\in X} p(x)log^{\frac{p(x)}{q(x)}}$

交叉熵(cross entropy) ：衡量估计模型和真实概率之间的差异

$H(X, q) = H(x) + D(p||q) = -\sum_x p(x)log(q(x))$

定义**语言$L = x_i \in p(x)$与其模型的交叉熵** 为：

交叉熵

$$H(L,q) = -lim_{n\to \inf} \frac{1}{n} \sum_{x_1^n} p(x_1^n) log(q(x_1^n))$$

$$H(L,q) = -\frac{1}{n} log(q(x_1^n))$$

迷惑度(perplexity) ：困惑的描述了当我们看到序列中每个单词后的惊讶程度，数值上为交叉熵的二次方，ppl数值更漂亮。

$$ppl = 2^{H(L,q)} = q(x_1^n)^{-\frac{1}{n}}$$

语言模型的构建方法

构建语言模型主要包括两类方法：

1、count-based N-gram语言模型

2、Neural N-gram 语言模型

3、RNN 语言模型

N-gram 模型：

非常简单，语言模型主要的起源，最成熟，依旧是语言模型最主要的解决办法。

对语言模型$p(w_1, w_2 .... w_N) = p(w_1)p(w_2|w_1)p(w_3|w_1, w_2)...p(w_N|w_1, w_2 .... w_{N-1})$做马尔科夫假设：

马尔科夫假设 ：下一个的出现仅仅依赖前边一个词或者几个词。（前边出现的词对下一个出现的词有约束能力（预测能力），条件依赖有一定局部性作用（越近影响越大），n-gram语言模型就是利用了语言的这个性质。）

（事实上这是不得已而采用的对参数空间进行裁剪对策，没有理论保证）

unigram ： 下一个的出现和前边出现的词没有关系：

$$p(I) = p(S) = p(w_1, w_2 ... w_n) = p(w_1)p(w_2)...p(w_n)$$

bigram ：下一个的出现仅仅依赖前边一个词

$$p(I) = p(S) = p(w_1, w_2 ... w_n) = p(w_1)p(w_2|w_1)...p(w_n|w_{n-1})$$

即使再简单的2-gram模型也会有很好的效果。

trigram ：下一个词出现的概率依赖于它前面的两个词：

$$p(I) = p(S) = p(w_1, w_2 ... w_n) = p(w_1)p(w_2|w_1)p(w_3|w_1, w_2)...p(w_n|w_{n-1}，w_{n-2})$$

语音识别领域，trigram表现比较良好。

参数估计

最大似然估计：

$$p(w_n|w_{n-N+1}, ... w_{n-1}) = \frac{C(w_{n-N+1} .... w_n)}{C(w_{n-N+1} .... w_{n-1})}$$

N-gram就是计算在大规模真实语料中统计n元词序列的个数。

从上图可以看出，随着参数的增加，参数空间将会迅速膨胀。

平滑处理

概率连乘，如果有一个概率为0，那么这个概率即为0，因此，需要做平滑处理，以保证不会出现概率为0情况。

下述是一个非常简单的线性插值方法，还有

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

s.t $\lambda_3 + \lambda_2 + \lambda_1 = 1$

还有许多其他复杂精妙的方法，这里就不再说明。

方法评价 ：

优点：1 、可以非常轻松处理大量数据，很容易在Mapreduce框架下完成，

2、具有强大的扩展能力。

3、训练很快

问题：

1、**N太大，参数空间过大，方差太大，泛化能力不佳**

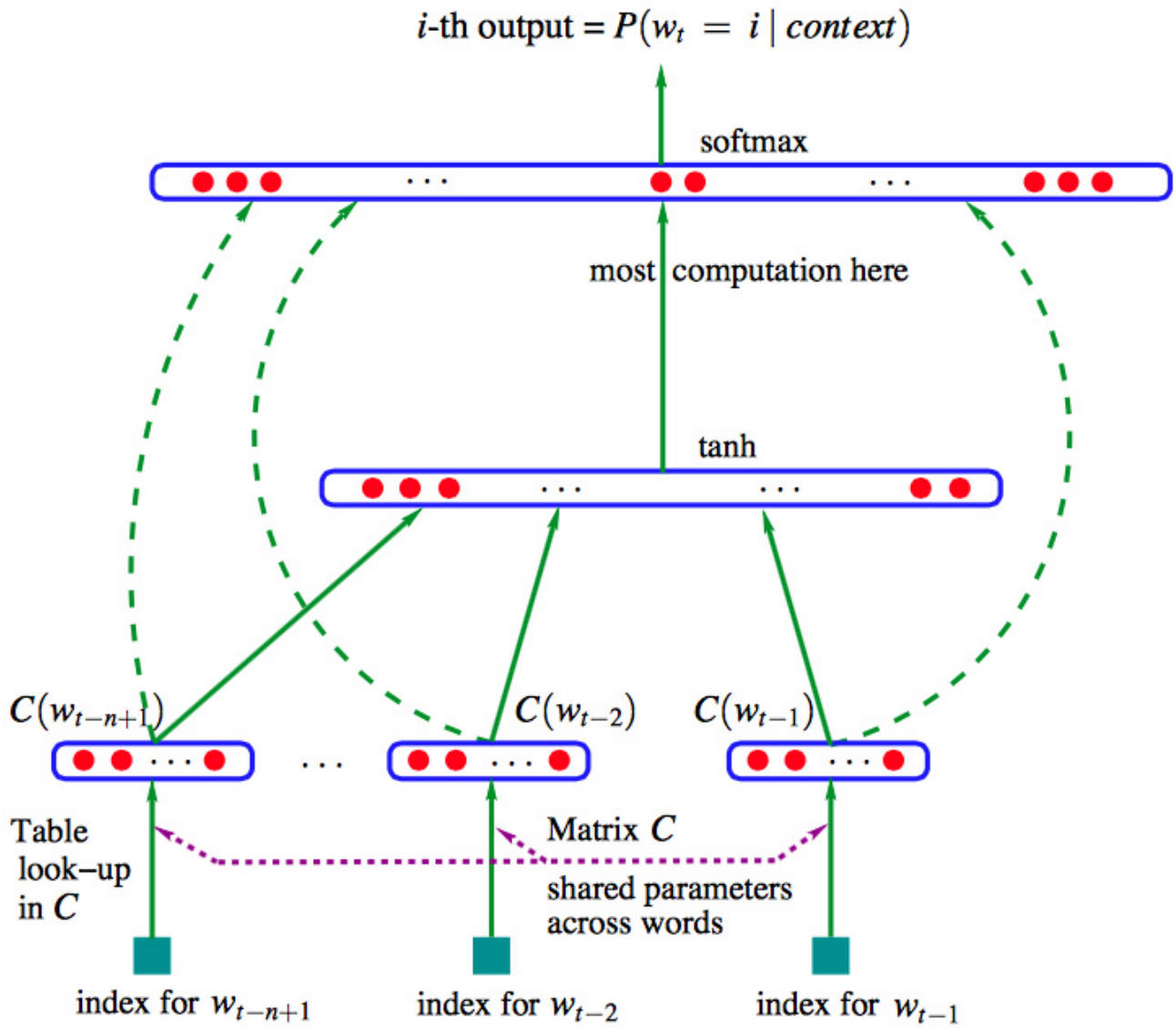
2、**不能捕捉更长远信息 。**

3、存在许多概率为0的分布（zipf law），**概率估计不准确** ，

4、马尔科夫假设，**概率估计有偏差 。**

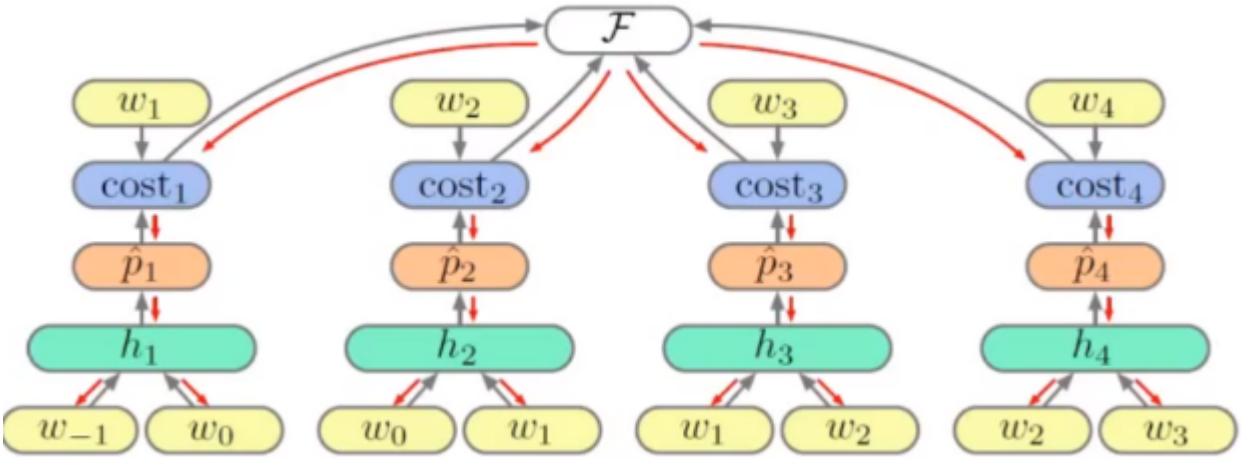
5、N-gram模型的本质就是把单词看作不同的符号，而没有考虑词与词之间的内部联系（相似性，gender等信息），**无法很好泛化。**

Neural N-gram 语言模型



Neureal N-gram 的网络架构如上图所示，输入为预测词前边N-1个词的one-hot vector（N对于N-gram选择的N），包含一个隐藏层，神经网络的输出为softmax之后每个词出现的概率（输出维度与词汇表长度一致），然后根据实际语料中的真实出现的下一个词的one-hot vector，通过最小化两者之间的交叉熵来训练模型。

目标函数：$F = -\frac{1}{N}\sum_n CE(w_n, \hat{p}_n)$



模型评价

优点 ：1、相比于基于统计的N-gram模型，Neural N-gram模型可以捕获更多的预测词与之前词之间的关系，对于未见过的词序列，可以**有更好的泛化能力**（可以更好的表达词与词之间的相似性）。

2、不需要存储大量的统计概率，因此，**需要更少的内存信息。**

3、仅仅通过目标函数将所有词联系起来，因此**非常容易并行**

缺点 ：1、Neural N-gram 模型只利用前N-1个词，预测下一个词，无法**捕捉长期依赖信息**

2、**无法利用统计的词频信息**

3、模型的参数数量受N的影响

RNN 语言模型

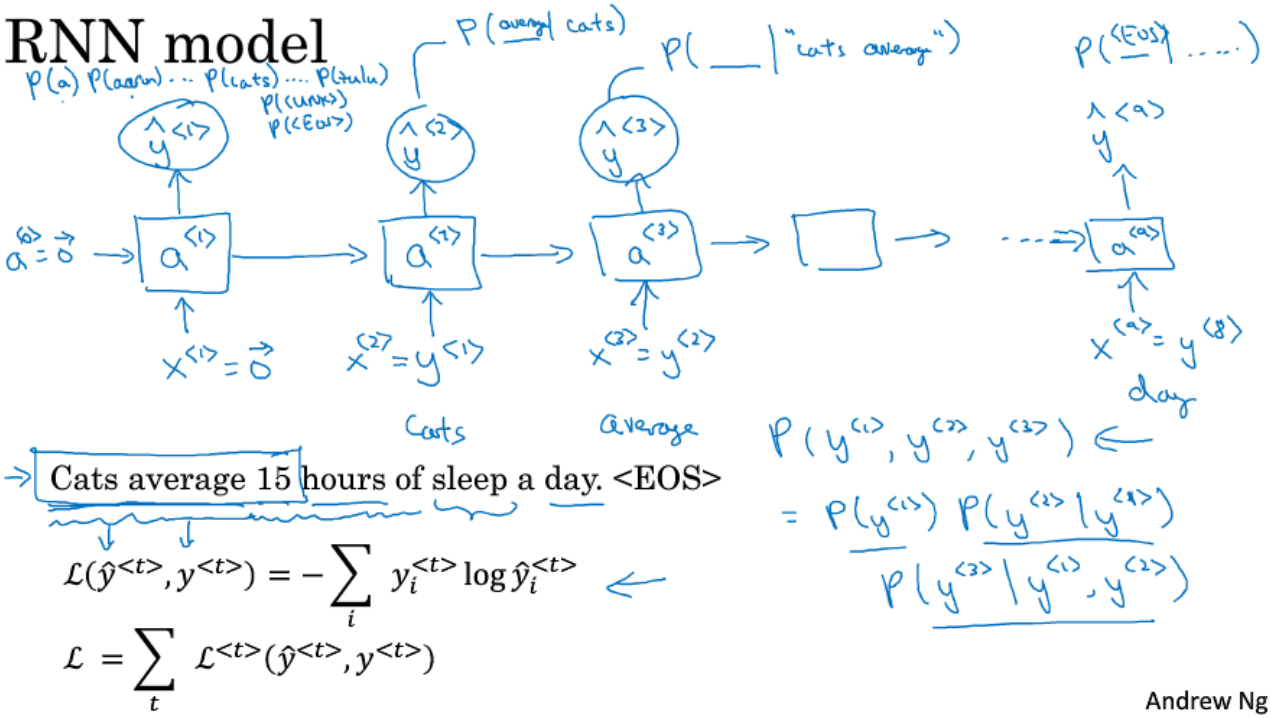
构造训练集 ：

1、收集一个包含一个很大的文本语料库，

2、对语料库中每个单词进行使用one-hot vector进行标记，增加作为句子的结束，对于不在词表中的词标记为，EOS和UNK都对应一个独立的one-hotvector。假设句子有N个单词，从而得到序列

($y^{<1>}, y^{<2>} \ldots . y^{<N>} y^{<N+1>}$), 其中 $y^{<N+1>}$ 对应EOS.

3、定义$x^{<t>} = y^{<t-1>}$ ，t=1时，$x^{<0>}$ 为零向量



RNN LM模型如上图所示，采用循环神经网络，每个时间点只输入一个单词one-hot vector$x^{<t>}$ ，并把前一步隐藏层中信息传递到下一个时间中作为输入（0时间点输入为零向量即可），输出为softmax归一化后每个单词出现的概率$\hat{y}^{<t>}$,reperence 为对应的$y^{<t>}$,在每个时间点计算$y^{<t>}$与$\hat{y}^{<t>}$的交叉熵来优化模型。

当需要计算序列（$y^{<1>}, y^{<2>} y^{<3>}$）出现的概率时，只需进行概率连乘即可，每个单项的概率即为对应的softmax对应的输出概率。比如第一个时间点softmax输出会给出$p(y^{<1>})$的概率，第二个softmax会给出$p(y^{<2>}|y^{<1>})$,第三个softmax会给出$p(y^{<3>}|y^{<1>}，y^{<2>})$

目标函数 ：

$J^t(\theta) = -\sum_{j=1}^{|V|} y_{t,j} log\hat{y}_{t,j}$

$J = -\frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{|V|} y_{t,j} log\hat{y}_{t,j}$

改进 ： 1、RNN可以采用LSTM或者GRU以获得更长期的信息，

2、可以构建更深层的循环神经网络（需要说明的是语言模型不需要Bi-direction RNN）。

训练 ：目标函数与Neural N-gram一样，也为交叉熵，但需要采用RNN的训练方法BPTT或者TBPTT。

模型评价

优点 ： 1、RNN 的核心是将历史压缩到隐藏层中，并传递到下一个时间点，因此该模型真正把我们从条件概率分布中解脱出来，从而**可能捕获更长时间的信息**

2、每次输入只有一个词，因此相比Neural N-gram模型，**需要更少的参数量，泛化性能更好**

3、将历史信息存储在隐藏层中，可以认为考虑了之前所有的信息，因此**概率计算是无偏的**

缺点 ： 1、常规的RNN模型由于**Vanish Gradient和Explode Gradient** 等问题，非常难训练

2、BPTT等训练算法**训练时间非常长**

3、与Neural N-gram模型一样，**无法利用统计的词频信息**

4、为了捕获更长期信息，隐藏层需要更大，但更大的隐藏层意味着更多的参数量，增加训练的难度和时间

字符级别的语言模型

上面几种语言模型都是基于词的语言模型（即词表中的词都是英语单词），根据应用的需要，我们还可以构建一个基于字符的语言模型，此时，词汇表中仅包含从a-z的字母，可能还有空格，数字0-9和一些其他符号，那么进行训练时，$y^{<1>}, y^{<2>}, \ldots y^{<N>}$ 为对应的字符，其他与基于词汇的RNN 语言模型相同。

优点 ：

1、不用担心出现位置的标识

2、不用面对巨大的softmax问题

3、可以捕获一定程度的字符结构和语法(英语中字符级别的结构(比如特点的前缀，后缀，词根等)可以表示一些句法和语义上的相关信息。

缺点 ：

使用字符对句子进行标记，字符级别的序列会非常长，所以将会面对**更严重的长期依赖的问题** ，**计算量也将更大** 。

目前主要的语言模型都是基于词汇的，但是随着计算力的不断增强，对于许多需要处理大量未知词汇，专有词汇的文本，采用基于字符的语言模型会有更好的效果。

相关课程：

哈工大 关毅 自然语言处理 语言模型

牛津大学&DeepMind Deep Learning for Natural Language Processing ,Language Modelling

斯坦福 CS224n:Natural Language Processing with Deep Learning ，Recurrent Neural Networks and Language Models

台湾大学 李宏毅 深度学习 语言模型

吴恩达 《深度学习工程师》 序列模型