语言模型

简介

语言模型：

简单地说，语言模型就是用来计算一个句子的概率的模型，也就是判断一句话是否合理的概率。

应用：

机器翻译，问答系统，语音识别，分词，输入法，搜索引擎的自动补全等也都应用到了语言模型。

语言模型建立：

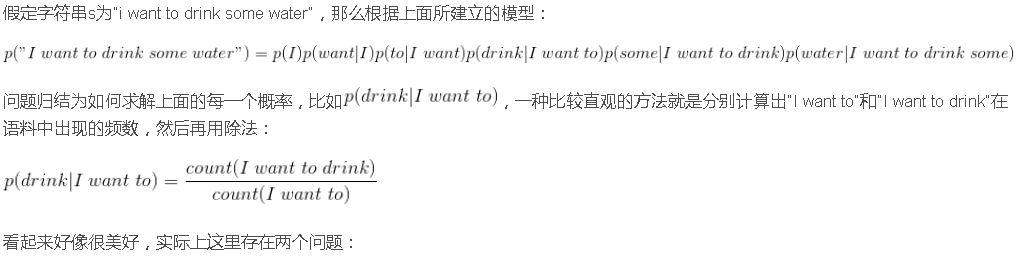
对于一个由T个词按顺序构成的句子，p(s)实际上求解的是字符串的联合概率，利用贝叶斯公式，链式分解如下：



从上面可以看到，一个统计语言模型可以表示成，给定前面的的词，求后面一个词出现的条件概率。

我们在求p(s)时实际上就已经建立了一个模型，这里的p(\*)就是模型的参数，如果这些参数已经求解得到，那么很容易就能够得到字符串s的概率。

求解问题：



（1）自由参数数目：

假定字符串中字符全部来自于大小为V的词典，上述例子中我们需要计算所有的条件概率，对于所有的条件概率，这里的w都有V种取值，那么实际上这个模型的自由参数数目量级是V^6，6为字符串的长度。

从上面可以看出，模型的自由参数是随着字符串长度的增加而指数级暴增的，这使我们几乎不可能正确的估计出这些参数。

（2）数据稀疏性：

从上面可以看到，每一个w都具有V种取值，这样构造出了非常多的词对，但实际中训练语料是不会出现这么多种组合的，那么依据最大似然估计，最终得到的概率实际是很可能是0。

解决方法：

基于规则的语言模型

**基于统计的语言模型- N-gram语言模型**

**基于神经网络的语言模型：DNN/RNN/LSTM**

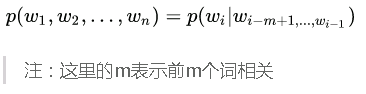
N-gram语言模型

马尔科夫假设：

马尔科夫假设是指，每个词出现的概率只跟它前面的少数几个词有关。比如，二阶马尔科夫假设只考虑前面两个词，相应的语言模型是三元模型。引入了马尔科夫假设的语言模型，也可以叫做马尔科夫模型。

马尔可夫链（Markov chain）为状态空间中经过从一个状态到另一个状态的转换的随机过程。该过程要求具备“无记忆”的性质：下一状态的概率分布只能由当前状态决定，在时间序列中它前面的事件均与之无关。

也就是说，应用了这个假设表明了当前这个词仅仅跟前面几个有限的词相关，因此也就不必追溯到最开始的那个词，这样便可以大幅缩减上述算式的长度。即式子变成了这样：



确定m的取值：

通常情况下，m的取值不能够太大，否则自由参数过多的问题依旧存在：

（1）当m=1时，即一个词的出现与它周围的词是独立，这种我们称为unigram，也就是一元语言模型，此时自由参数量级是词典大小V。

（2）当m=2时，即一个词的出现仅与它前面的一个词有关时，这种我们称为bigram，叫二元语言模型，也叫一阶马尔科夫链，此时自由参数数量级是V^2。

（3）当m=3时，即一个词的出现仅与它前面的两个词有关，称为trigram，叫三元语言模型，也叫二阶马尔科夫链，此时自由参数数量级是V^3。

一般情况下只使用上述取值，因为从上面可以看出，自由参数的数量级是n取值的指数倍。

从模型的效果来看，理论上m的取值越大，效果越好。但随着m取值的增加，效果提升的幅度是在下降的。同时还涉及到一个可靠性和可区别性的问题，参数越多，可区别性越好，但同时单个参数的实例变少从而降低了可靠性。

N-gram语言模型的求解跟传统统计语言模型一致，都是求解每一个条件概率的值，简单计算N元语法在语料中出现的频率，然后归一化。

平滑化：

我们在传统统计语言模型提出了两个问题：自由参数数目和数据稀疏，上述N-gram只是解决了第一个问题，而平滑化就是为了解决第二个问题。

假设有一个词组在训练语料中没有出现过，那么它的频次就为0，但实际上能不能认为它出现的概率为0呢？显然不可以，我们无法保证训练语料的完备性。那么，解决的方法是什么？如果我们默认每一个词组都出现1次呢，无论词组出现的频次是多少，都往上加1，这就能够解决概率为0的问题了。

上述的方法就是加1平滑，也称为拉普拉斯平滑。平滑化还有许多方法，这里就不展开介绍了：

（1）加法平滑

（2）古德-图灵平滑

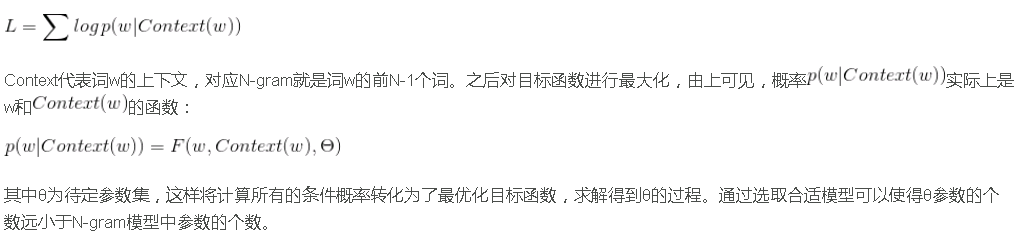
（3）K平滑

神经概率语言模型

在N-gram语言模型中，计算条件概率的方法是简单的用词频做除法然后归一化。

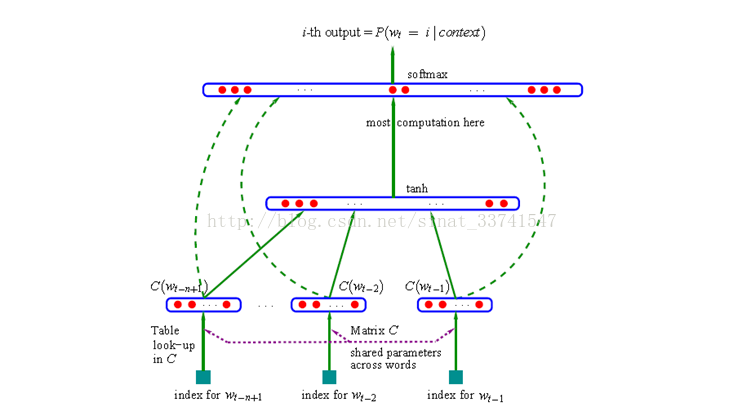
在机器学习的领域中，通用的做法是：对所考虑的问题建模后先为其构造一个目标函数，然后对这个目标函数进行优化，从而求得一组最优的参数，最后再利用这组参数对应的模型来进行预测。

那么在上述的语言模型中，利用最大化对数似然，将目标函数设为：



Begio等人在2003年发表的A Neural Probabilistic Language Model，里面详解了这个方法。

基本的思想其实与上述的前置知识有所联系，既然是神经概率语言模型，那么实现的时候自然有一个神经网络，结构图如下：



它包括了四个层：输入层、投影层、隐藏层和输出层。

计算流程：

（1）输入层

这里就是词w的上下文，如果用N-gram的方法就是词w的前n-1个词了。每一个词都作为一个长度为V的one-hot向量传入神经网络中

（2）投影层

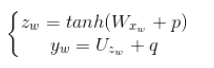
在投影层中，存在一个look-up表C，C被表示成一个V\*m的自由参数矩阵，其中V是词典的大小，而m作为自定义的参数，一般是10^2的倍数。

表C中每一行都作为一个词向量存在，这个词向量可以理解为每一个词的另一种分布式表示。每一个one-hot向量都经过表C的转化变成一个词向量。

n-1个词向量首尾相接的拼起来，转化为(n-1)m的列向量输入到下一层。

（3）隐藏层、输出层

之后再对列向量进行计算，大致如下：



其中tanh是激活函数，是为归一化的log概率，之后再用softmax进行归一化，就得到最终的概率输出了。

在前置知识中我们提到了参数θ，那么在神经网络中，实际的参数如下：

词向量：v(w)，w以及填充向量

神经网络参数：W，p，U，q

在传统统计语言模型中，我们提出两个问题：自由参数数目和数据稀疏。

这里在实际上使用参数θ代替了自由参数指数级的求解，而数据稀疏问题，我们在最后使用softmax进行归一化，求解出来的概率是平滑的，所以也解决了这个问题。

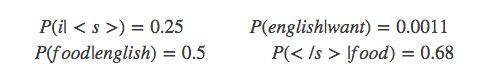
N-Gram经典应用

利用N-Gram模型评估语句是否合理：

假设现在有一个语料库，我们统计了下面的一些词出现的数量

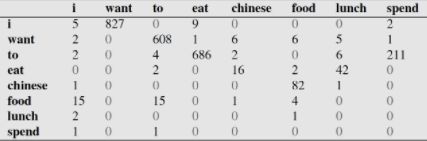
https://pic1.zhimg.com/80/v2-43745e5526b6eb8341a896c8abe8c640_hd.jpg

下面的这些概率值作为已知条件：

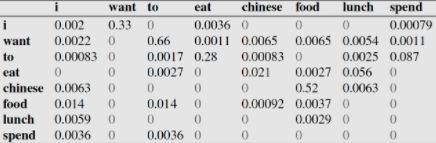


p(want|<s>) = 0.25

下面这个表给出的是基于Bigram模型进行计数之结果



例如，其中第一行，第二列 表示给定前一个词是 “i” 时，当前词为“want”的情况一共出现了827次。据此，我们便可以算得相应的频率分布表如下。



比如说，我们就以表中的p(eat|i)=0.0036这个概率值讲解，从表一得出“i”一共出现了2533次，而其后出现eat的次数一共有9次，p(eat|i)=p(eat,i)/p(i)=count(eat,i)/count(i)=9/2533 = 0.0036

下面我们通过基于这个语料库来判断s1=“<s> i want english food</s>” 与s2 = "<s> want i english food</s>"哪个句子更合理：

首先来判断p(s1)

P(s1)=P(i|<s>)P(want|i)P(english|want)P(food|english)P(</s>|food)

=0.25×0.33×0.0011×0.5×0.68=0.000031

再来求p(s2)？

P(s2)=P(want|<s>)P(i|want)P(english|want)P(food|english)P(</s>|food)

=0.25\*0.0022\*0.0011\*0.5\*0.68 = 0.00000002057

通过比较我们可以明显发现0.00000002057<0.000031,也就是说s1= "i want english food</s>"更合理。

**n-gram 模型其他应用举例**

n-gram模型也有其他很多应用，以下一一举例：

1.**研究人类文明：**n-gram模型催生了一门新学科([Culturomics](http://link.zhihu.com/?target=https%3A//en.wikipedia.org/wiki/Culturomics))的成立，通过数字化的文本，来研究人类行为和文化趋势。《可视化未来》这本书有详细介绍，也可以通过知乎上的[详细介绍](https://www.zhihu.com/question/26166417)，还有就是TED上的视频：[what\_we\_learned\_from\_5\_million\_books](http://link.zhihu.com/?target=http%3A//www.ted.com/talks/what_we_learned_from_5_million_books) 。

2.**搜索引擎：**当你在谷歌或者百度的时候，输入一个或几个词，搜索框通常会以下拉菜单的形式给出几个像下图一样的备选，这些备选其实是在猜想你想要搜索的那个词串。如下图：



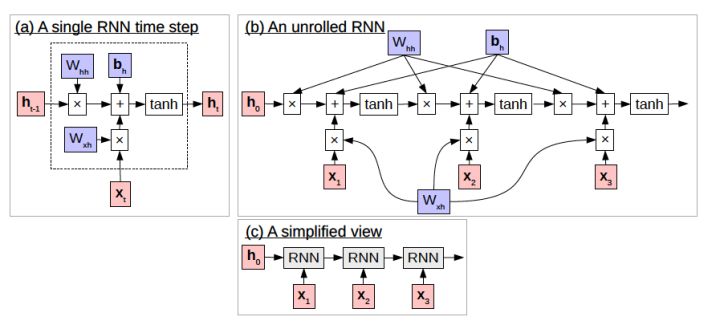
实际上这些都是根据语言模型得出。比如使用的是二元语言模型预测下一个单词：排序的过程就是：

p(”不一样“|"我们")>p(”的爱“|"我们")>p(”相爱吧“|"我们")>.......>p("这一家"|”我们“)，这些概率值的求法和上面提到的完全一样，数据的来源可以是用户搜索的log。

3.**输入法：**比如输入“zhongguo”，可能的输出有：中国，种过，中过等等....这背后的技术就要用到n-gram语言模型了。item就是每一个拼音对应的可能的字。

基于RNN的语言模型

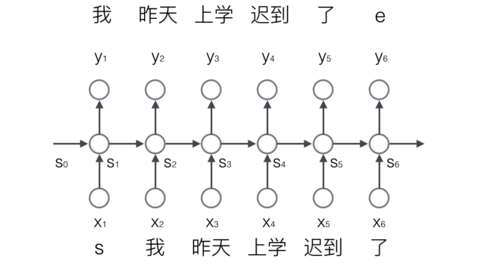
好的语言模型应当至少捕获自然语言的两个特征：语法特性与语义特性。为了保证语法的正确性，我们往往只需要考虑生成词的前置上下文；这也就意味着语法特性往往是属于局部特性。而语义的一致性则复杂了许多，我们需要考虑大量的乃至于整个文档语料集的上下文信息来获取正确的全局语义**。**神经网络语言模型相较于经典的 N-gram 模型具有更强大的表现力与更好的泛化能力，不过传统的 N-gram 语言模型与 [Bengio et al., 2003] 中提出的神经网络语言模型都不能有效地捕获全局语义信息。为了解决这个问题，[Mikolov et al., 2010; 2011] 中提出的基于循环神经网络（Recurrent Neural Network, RNN）的语言模型使用了隐状态来记录词序的历史信息，其能够捕获语言中的长程依赖。在自然语言中，往往在句式中相隔较远的两个词却具备一定的语法与语义关联，譬如He doesn't have very much confidence in himself 与 She doesn't have very much confidence in herself 这两句话中的<He, himself>与<She, herself>这两个词对，尽管句子中间的词可能会发生变化，但是这两种词对中两个词之间的关联却是固定的。这种依赖也不仅仅出现在英语中，在汉语、俄罗斯语中也都存在有大量此类型的词对组合。而另一种长期依赖（Long-term Dependencies）的典型就是所谓的选择限制（Selectional Preferences）;简而言之，选择限制主要基于已知的某人会去做某事这样的信息。譬如我要用叉子吃沙拉与我要和我的朋友一起吃沙拉这两句话中，叉子指代的是某种工具，而我的朋友则是伴侣的意思。如果有人说我要用双肩背包来吃沙拉就觉得很奇怪了，双肩背包并不是工具也不是伴侣；如果我们破坏了这种选择限制就会生成大量的无意义句子。最后，某个句式或者文档往往都会归属于某个主题下，如果我们在某个技术主题的文档中突然发现了某个关于体育的句子，肯定会觉得很奇怪，这也就是所谓的破坏了主题一致性。



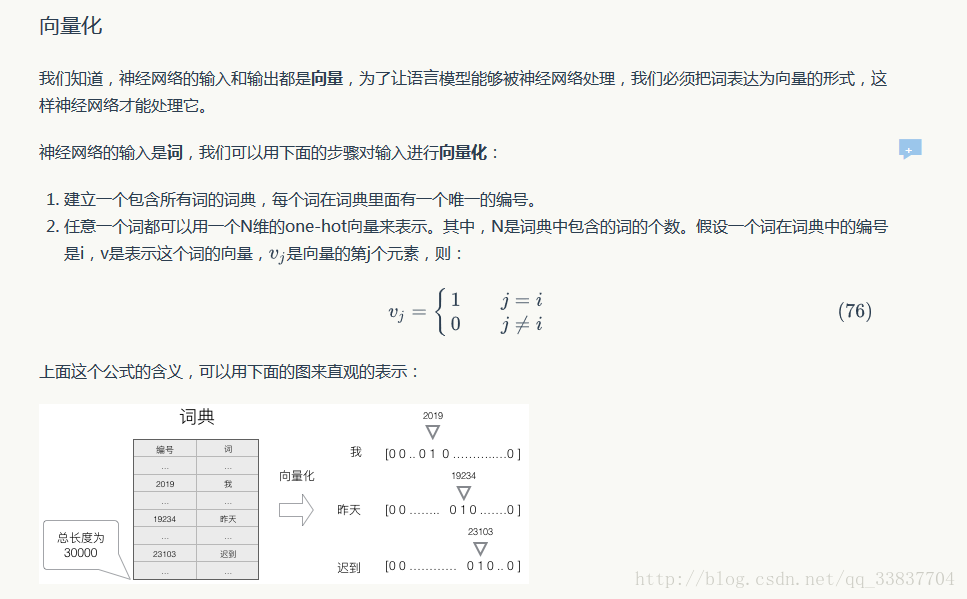
首先把词依次输入到循环神经网络中，每输入一个词，循环神经网络就输出截止到目前为止，下一个最可能的词。例如，当我们依次输入：

我 昨天 上学 迟到 了

神经网络的输出如下图所示：

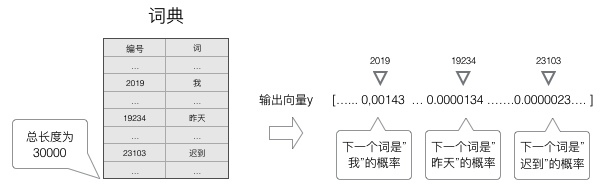


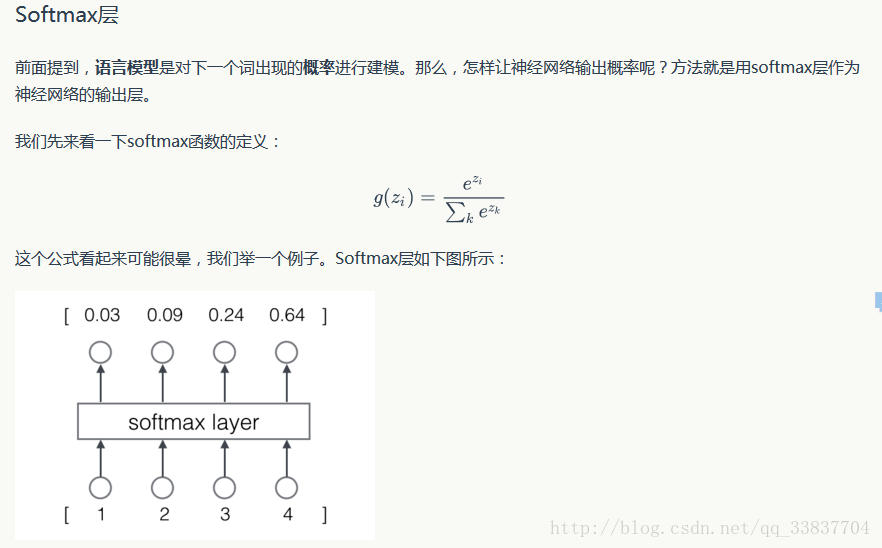
其中，s和e是两个特殊的词，分别表示一个序列的开始和结束。

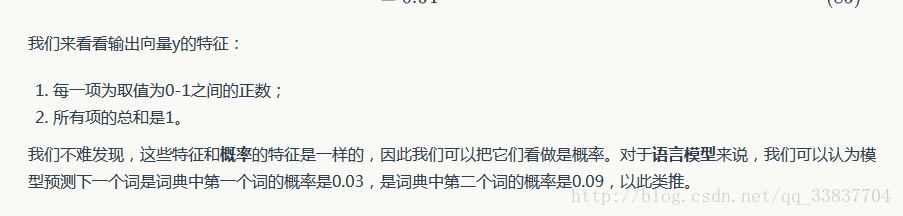


使用这种向量化方法，我们就得到了一个高维、**稀疏**的向量（稀疏是指绝大部分元素的值都是0）。处理这样的向量会导致我们的神经网络有很多的参数，带来庞大的计算量。因此，往往会需要使用一些降维方法，将高维的稀疏向量转变为低维的稠密向量。不过这个话题我们就不再这篇文章中讨论了。

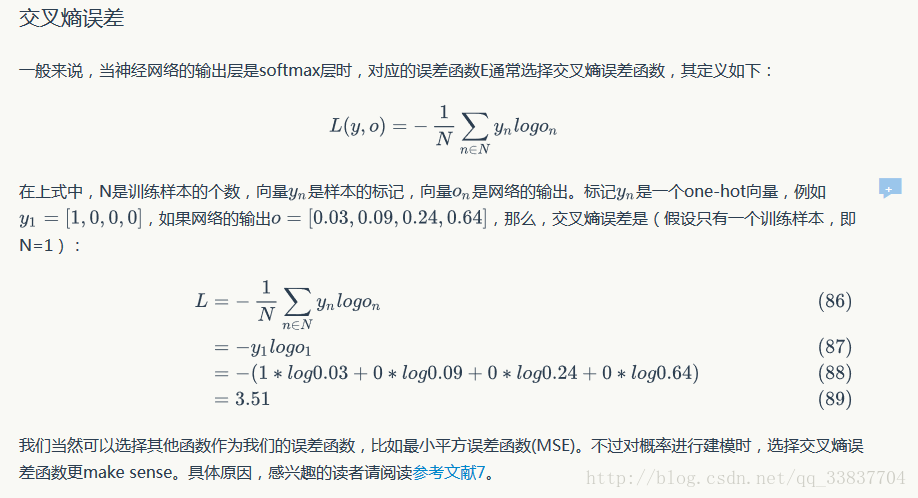
语言模型要求的输出是下一个最可能的词，我们可以让循环神经网络计算计算词典中每个词是下一个词的概率，这样，概率最大的词就是下一个最可能的词。因此，神经网络的输出向量也是一个N维向量，向量中的每个元素对应着词典中相应的词是下一个词的概率。如下图所示：











参考文献：

<https://zhuanlan.zhihu.com/p/34219483>

<https://blog.csdn.net/qq_36330643/article/details/80143960>

<https://blog.csdn.net/qq_33837704/article/details/79513049>