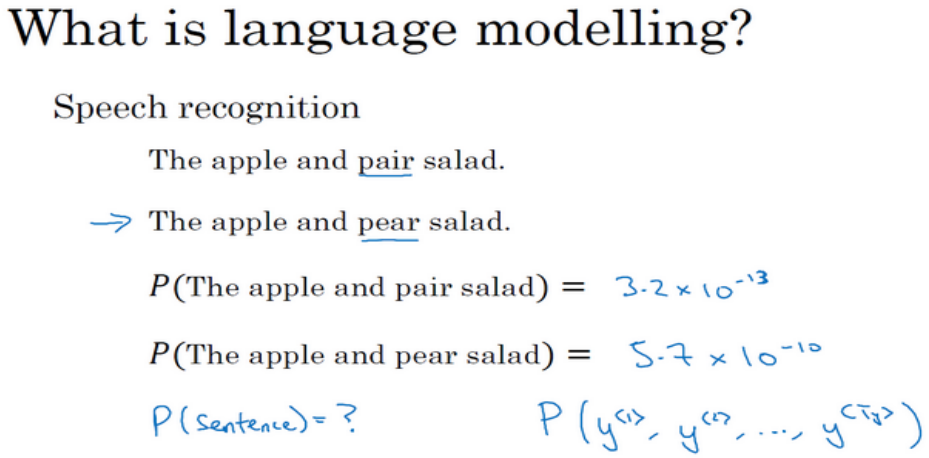
1.6 语言模型和序列生成（Language model and sequence generation）

在自然语言处理中，构建语言模型是最基础的也是最重要的工作之一，并且能用 **RNN** 很好地实现。



所以什么是语言模型呢？比如你在做一个语音识别系统，你听到一个句子，**“the apple and pear（pair） salad was delicious.”**，所以我究竟说了什么？我说的是 **“the apple and pair salad”**，还是**“the apple and pear salad”**？（**pear** 和 **pair** 是近音词）。你可能觉得我说的应该更像第二种，事实上，这就是一个好的语音识别系统要帮助输出的东西，即使这两句话听起来是如此相似。**而让语音识别系统去选择第二个句子的方法就是使用一个语言模型，他能计算出这两句话各自的可能性。**

举个例子，一个语音识别模型可能算出第一句话的概率是:

$P(\text{The apple and pair salad}) = 3.2 \times 10^{-13}$，

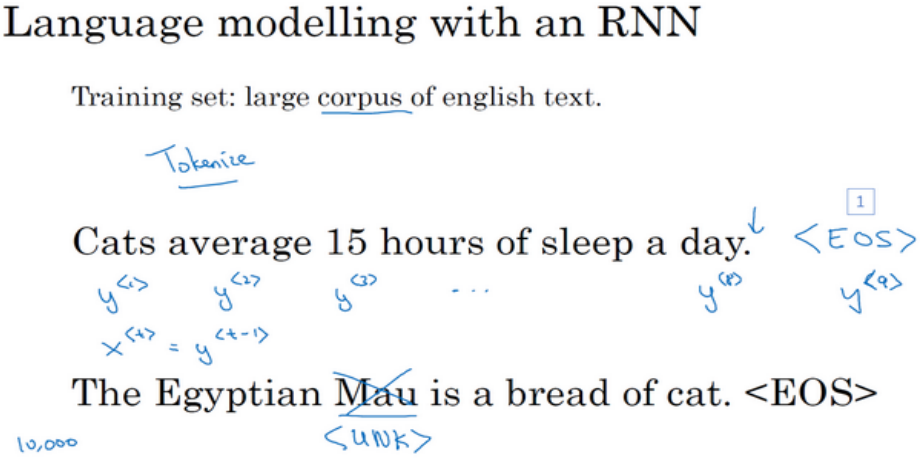
而第二句话的概率是$P(\text{The apple and pear salad}) = 5.7 \times 10^{-10}$，

比较这两个概率值，显然我说的话更像是第二种，因为第二句话的概率比第一句高出 1000 倍以上，这就是为什么语音识别系统能够在这两句话中作出选择。

所以语言模型所做的就是，它会告诉你某个特定的句子它出现的概率是多少，根据我所说的这个概率，假设你随机拿起一张报纸，打开任意邮件，或者任意网页或者听某人说下一句话，并且这个人是你的朋友，这个你即将从世界上的某个地方得到的句子会是某个特定句子的概率是多少，例如**“the apple and pear salad”**。它是两种系统的基本组成部分，一个刚才所说的语音识别系统，还有机器翻译系统，它要能正确输出最接近的句子。而语言模型做的

最基本工作就是输入一个句子，准确地说是一个文本序列，$y^{<1>}$，$y^{<2>}$一直到$y^{<T_y>}$。对于语言模型来说，用y来表示这些序列比用x来表示要更好，然后语言模型会估计某个句子序列中各个单词出现的可能性。

那么如何建立一个语言模型呢？为了使用 **RNN** 建立出这样的模型，你首先需要一个训练集，包含一个很大的英文文本语料库（**corpus**）或者其它的语言，你想用于构建模型的语言的语料库。语料库是自然语言处理的一个专有名词，意思就是很长的或者说数量众多的英文句子组成的文本。

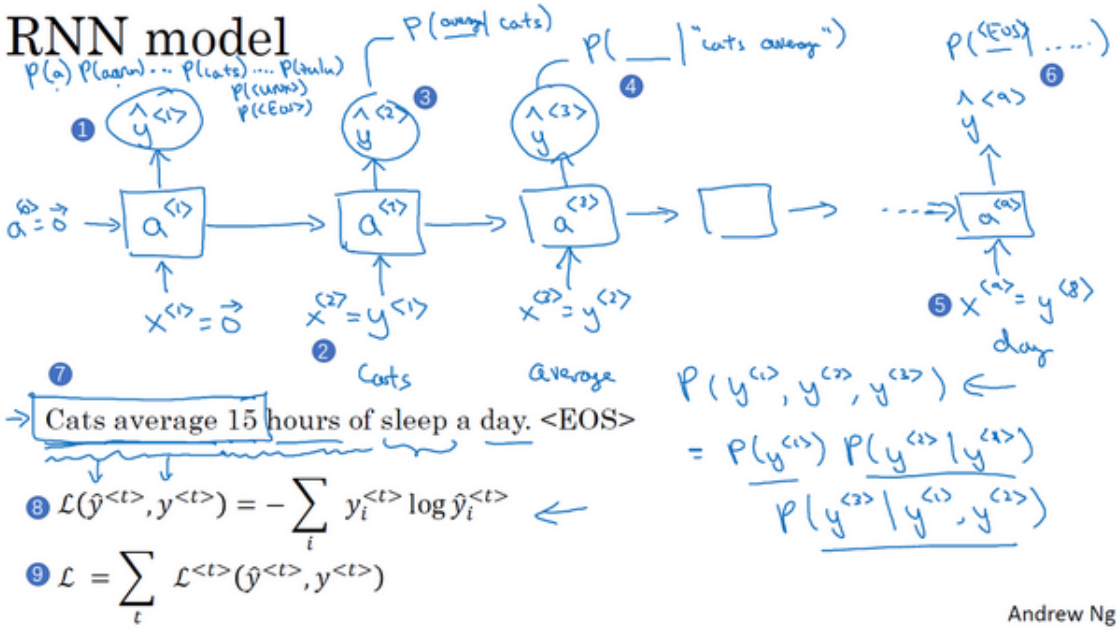


假如说，你在训练集中得到这么一句话，“**Cats average 15 hours of sleep a day.**”(猫一天睡 15 小时)，你要做的第一件事就是将这个句子标记化，意思就是像之前视频中一样，建立一个字典，然后将每个单词都转换成对应的 **one-hot** 向量，也就是字典中的索引。可能还有一件事就是你要定义句子的结尾，一般的做法就是增加一个额外的标记，叫做 **EOS**（上图编号 1 所示），它表示句子的结尾，这样能够帮助你搞清楚一个句子什么时候结束，我们之后会详细讨论这个。**EOS** 标记可以被附加到训练集中每一个句子的结尾，如果你想要你的模型能够准确识别句子结尾的话。在本周的练习中我们不需要使用这个 **EOS** 标记，不过在某些应用中你可能会用到它，不过稍后就能见到它的用处。于是在本例中我们，如果你加了 **EOS** 标记，这句话就会有 9 个输入，有$y^{<1>}$，$y^{<2>}$一直到$y^{<9>}$。在标记化的过程中，你可以自行决定要不要把标点符号看成标记，在本例中，我们忽略了标点符号，所以我们只把 **day** 看成标志，不包括后面的句号，如果你想把句号或者其他符号也当作标志，那么你可以将句号也加入你的字典中。

现在还有一个问题如果你的训练集中有一些词并不在你的字典里，比如说你的字典有 10,000 个词，10,000 个最常用的英语单词。现在这个句，“**The Egyptian Mau is a bread of cat.**”其中有一个词 **Mau**，它可能并不是预先的那 10,000 个最常用的单词，在这种情况下，你可

以把 **Mau** 替换成一个叫做 **UNK** 的代表未知词的标志，我们只针对 **UNK** 建立概率模型，而不是针对这个具体的词 **Mau**。

完成标识化的过程后，这意味着输入的句子都映射到了各个标志上，或者说字典中的各个词上。下一步我们要构建一个 **RNN** 来构建这些序列的概率模型。在下一张幻灯片中会看到的一件事就是最后你会将$x^{<t>}$设为$y^{<t-1>}$。



现在我们来建立 **RNN** 模型，我们继续使用“**Cats average 15 hours of sleep a day.**”这个句子来作为我们的运行样例，我将会画出一个 **RNN** 结构。在第 0 个时间步，你要计算激活项$a^{<1>}$，它是以$x^{<1>}$作为输入的函数，而$x^{<1>}$会被设为全为 0 的集合，也就是 0 向量。在之前的$a^{<0>}$按照惯例也设为 0 向量，于是$a^{<1>}$要做的就是它会通过 **softmax** 进行一些预测来计算出第一个词可能会是什么，其结果就是$\hat{y}^{<1>}$（上图编号 1 所示），这一步其实就是通过一个 softmax 层来预测字典中的任意单词会是第一个词的概率，比如说第一个词是a的概率有多少，第一个词是 **Aaron** 的概率有多少，第一个词是 **cats** 的概率又有多少，就这样一直到 **Zulu** 是第一个词的概率是多少，还有第一个词是 **UNK**（未知词）的概率有多少，还有第一个词是句子结尾标志的概率有多少，表示不必阅读。所以$\hat{y}^{<1>}$的输出是 softmax 的计算结果，它只是预测第一个词的概率，而不去管结果是什么。在我们的例子中，最终会得到单词 **Cats**。所以 **softmax** 层输出 10,000 种结果，因为你的字典中有 10,000 个词，或者会有 10,002 个结果，因为你可能加上了未知词，还有句子结尾这两个额外的标志。

然后 **RNN** 进入下个时间步，在下一时间步中，仍然使用激活项$a^{<1>}$，在这步要做的是

计算出第二个词会是什么。现在我们依然传给它正确的第一个词，我们会告诉它第一个词就是 **Cats**，也就是$\hat{y}^{<1>}$，告诉它第一个词就是 **Cats**，这就是为什么$y^{<1>} = x^{<2>}$（上图编号 2 所示）。然后在第二个时间步中，输出结果同样经过 **softmax** 层进行预测，**RNN** 的职责就是预测这些词的概率（上图编号 3 所示），而不会去管结果是什么，可能是 b 或者 **arron**，可能是 **Cats** 或者 **Zulu** 或者 **UNK**（未知词）或者 **EOS** 或者其他词，它只会考虑之前得到的词。所以在这种情况下，我猜正确答案会是 **average**，因为句子确实就是 **Cats average** 开头的。

然后再进行 **RNN** 的下个时间步，现在要计算$a^{<3>}$。为了预测第三个词，也就是 15，我们现在给它之前两个词，告诉它 **Cats average** 是句子的前两个词，所以这是下一个输入，$x^{<3>} = y^{<2>}$，输入 **average** 以后，现在要计算出序列中下一个词是什么，或者说计算出字典中每一个词的概率（上图编号 4 所示），通过之前得到的 **Cats** 和 **average**，在这种情况下，正确结果会是 15，以此类推。

一直到最后，没猜错的话，你会停在第 9 个时间步，然后把$x^{<9>}$也就是$y^{<8>}$传给它（上图编号 5 所示），也就是单词 **day**，这里是$a^{<9>}$，它会输出$y^{<9>}$，最后的得到结果会是 **EOS** 标志，在这一步中，通过前面这些得到的单词，不管它们是什么，我们希望能预测出 **EOS** 句子结尾标志的概率会很高（上图编号 6 所示）。

所以 **RNN** 中的每一步都会考虑前面得到的单词，比如给它前 3 个单词（上图编号 7 所示），让它给出下个词的分布，这就是 **RNN** 如何学习从左往右地每次预测一个词。

接下来为了训练这个网络，我们要定义代价函数。于是，在某个时间步t，如果真正的词是$y^{<t>}$，而神经网络的 **softmax** 层预测结果值是$y^{<t>}$，那么这（上图编号 8 所示）就是 **softmax** 损失函数，$L\left(\hat{y}^{<t>}, y^{<t>}\right) = -\sum_i y_i^{<t>} \log \hat{y}_i^{<t>}$。而总体损失函数（上图编号 9 所示）$L = \sum_t L^{<t>}\left(\hat{y}^{<t>}, y^{<t>}\right)$，也就是把所有单个预测的损失函数都相加起来。

$$P(y^{<1>}, y^{<2>}, y^{<3>}) \leftarrow$$
$$= \underbrace{P(y^{<1>})}_{①} \underbrace{P(y^{<2>} \mid y^{<1>})}_{②} \underbrace{P(y^{<3>} \mid y^{<1>}, y^{<2>})}_{③}$$

如果你用很大的训练集来训练这个 **RNN**，你就可以通过开头一系列单词像是 **Cars average 15** 或者 **Cars average 15 hours of** 来预测之后单词的概率。现在有一个新句子，它是$y^{<1>}$，$y^{<2>}$，$y^{<3>}$，为了简单起见，它只包含 3 个词（如上图所示），现在要计算出整个句子中各个单词的概率，方法就是第一个 softmax 层会告诉你$y^{<1>}$的概率（上图编号 1 所示），

这也是第一个输出，然后第二个 softmax 层会告诉你在考虑$y^{<1>}$的情况下$y^{<2>}$的概率（上图编号 2 所示），然后第三个 softmax 层告诉你在考虑$y^{<1>}$和$y^{<2>}$的情况下$y^{<3>}$的概率（上图编号 3 所示），把这三个概率相乘，最后得到这个含 3 个词的整个句子的概率。

这就是用 **RNN** 训练一个语言模型的基础结构，可能我说的这些东西听起来有些抽象，不过别担心，你可以在编程练习中亲自实现这些东西。下一节课用语言模型做的一件最有趣的事就是从模型中进行采样。