三、词向量模型

word2vec 是 Google 于 2013 年开源推出的一个用于获取 word vector 的工具包，它简单、高效，因此引起了很多人的关注。主要思想是2013年Tomas Mikolov 发表的两篇相关的论文 [1][2]。word2vec（word to vector）顾名思义，这是一个将单词转换成向量形式的工具。通过转换，可以把对文本内容的处理简化为向量空间中的向量运算，计算出向量空间上的相似度，来表示文本语义上的相似度。通常在对文本处理算法中可以先把词转换为连续向量，替代词在语料库中的one-hot表示，再去输入算法模型中去计算处理。word2vec为计算向量词提供了有效的连续词袋（bag-of-words）和skip-gram两种架构实现。接下来我们就来看看它是怎样实现的，剖析其内部数学原理，深入理解word2vec所蕴含的算法思想。

**3.1 预备知识**

本节主要介绍word2vec的相关重要知识，包括sigmod函数,逻辑回归,Beyes公式和Huffman编码等。

**3.1.1 sigmod函数**

sigmod函数是神经网络中常用的激活函数之一，其定义为：

该函数定义域为()，值域为(0,1).其图像如下图所示：



其函数的具体求导形式如下：

可得到和的导数为：

**3.1.2 逻辑回归**

二分类问题在分类问题中比较常见。例如，垃圾邮件分类，对于某个用户是否可以放贷，验证码操作是否是机器人等。假设为一个二分类问题中的一个数据样本，其中，。当的时候样本为正例，的时候为负例。

利用sigmod函数，对任意的样本,可以将二分类问题的hypothesis函数写作：

上式中的为待定参数，为了符号上的简化，引入,将扩展为,还将其仍记为。于是就可以写为：

取阈值T=0.5，二分类的判别公式为：

上面的参数是怎么来设定？当然是根据损失函数来求得。我们取对数**损失函**数形式如下：

对于上面的分段函数我们也可以写成一个式子：

**3.1.3 Bayes公式**

贝叶斯定理由英国数学家贝叶斯 ( Thomas Bayes 1702-1761 ) 发展，用来描述两个条件概率之间的关系，如果记和分别表示事件A和事件B的发生概率，表示在事件发生的情况下事件发生概率。表示事件和事件同时发生的概率。贝叶斯公式表示如下：

结合上式可以得到：

**3.1.4 Huffman树**

树是重要的数据结构，它是数据元素(结点)按照分支关系组织起来，若干的不相交的树所构成的集合称为森林。

* **路径和路径长度：**

在一棵树中，从一个结点往下可达到的孩子或孙子的结之间的通路，称为路径，通路中的分支数目称为路径长度，若规定根结点为第1层，则根结点第L层结点的路径长度为L-1。

* **结点的权值和带权路径**：

若为树中的每个结点赋予一个具有某种含义的数值(非负)，则这个数值称为该结点的权值。结点的带权路径长度是指从根结点到该结点的路径长度与该结点的权值乘积。

* 二叉树：

二叉树是指每个节点最多有两个子树的有序树。两个子树分别称为“左子树”和右子树。定义中的有序是指两个子树有左右之分，不能颠倒顺序。

给定个权值和个叶子结点，构造一棵二叉树，使它的带权路径长度达到最小，则这样的二叉树为最优二叉树，也叫Huffman树。

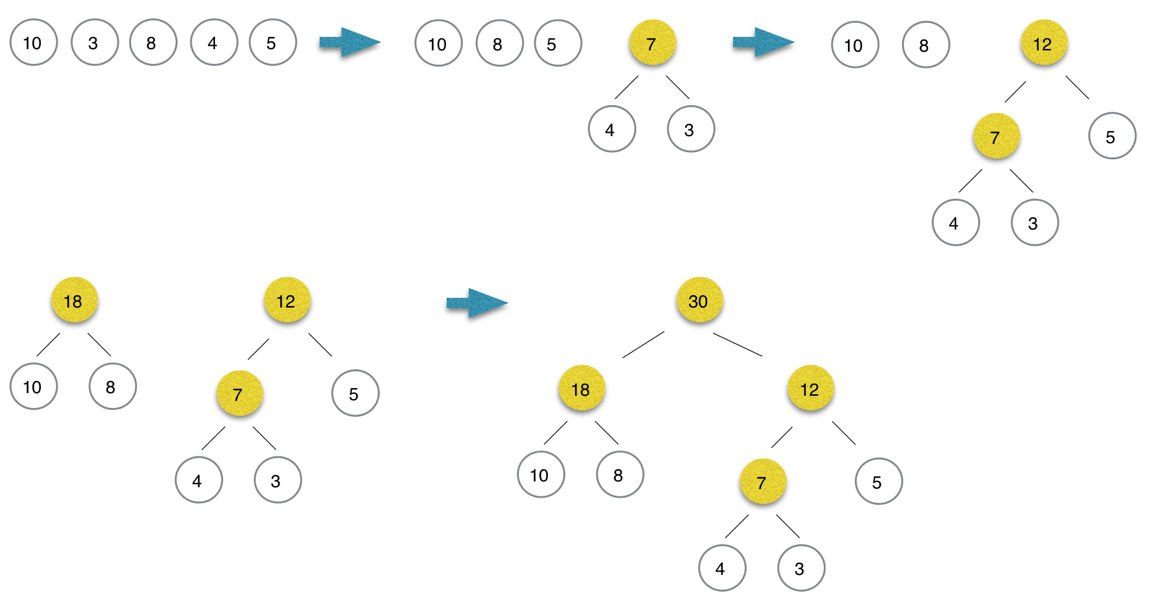
**3.1.5 Huffman树的构造**

给定个权值作为二叉树的叶子结点，可以用以下的方式来构造Huffman树。

1. 将看作一个有棵树的森林(每棵树有一个结点)
2. 在森林中选出两个根结点权重最小的树合并，作为一棵新树的左，右子树，且新树的根结点权重为其左右子树根集结点权重之和。
3. 从森林中删除选取的两棵子树，并将新树加入森林。
4. 重复(2)(3),直到森林中只剩下一棵树为止，该树就是所需要的Huffman树。

下面有一个具体实例：

假设在关于2018年俄罗斯世界杯的一篇报道中，经过统计，“我”，“喜欢”，“法国队”，“世界杯”，“夺冠”。这五个词的出现次数分别是：10，3，8，4，5。这样构造Huffman树的过程如下：



算法过程如上图所示，在最终构造的Huffman树中**词频越大的结点距离根根结点越近**。黄色的结点是新增的结点，如果叶子结点为个，则新增的节点数为个。在上图中我们默认在结点合并中权值大的结点放在左子树。

**3.1.4 Huffman编码**

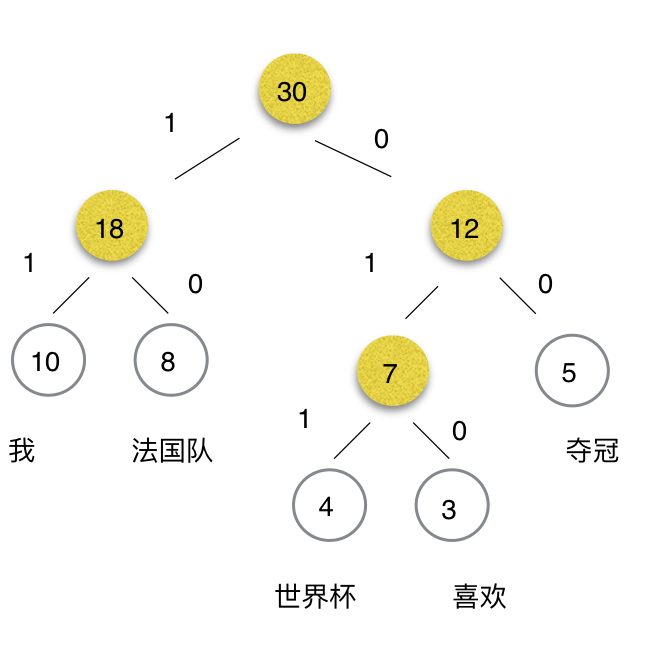
在数据通信中，需要将传送的文字转换成二进制字符串，用0/1码不同排列来表示字符。例如：需要传送的文本为：“AFTER DATA EAR ARE ART AREA”,在这，个字符集为“A，E，R，T，F，D”。各个字符出现的频率为8，4，5，3，1，1。现在要为这几个字符设计编码。

要区别6个字符最简单的二进制编码方式是等长编码，固定采用三位bit位，可以用000，001，010，011，100，101，对这6个字符编码。当对方接受到传送的二进制字符串后，在按照3位切分，之后译码转换为字符本身即可。

可知编码的长度取决于文本集中不同的字符个数，如果报文中的不同字符个数为26个，则需要的编码长度为5位，25=32>26。然而通信中传送总是希望长度越短越好。在现实中各个字符出现的频率或使用次数是不同的，比如A，B，C的使用频率要大于Z，X，Y。自然会想到让频率高的字符设计成短码，使用频率低的字符设计位长码，这样来优化整个报文的编码。

为了使得不等长的编码为前缀编码(要求一个字符的编码不能为另一个编码的前缀)，可以用每一个字符作为叶子结点生成一棵二叉编码树，为了获得传送报文最短，可以将每个字符出现的频率作为权值赋给每一个叶子结点。于是实现了频率小编码长，频率大编码短。

下图给出了在上面例子中的五个词的编码情况：“我”：11，“法国队”：10，“世界杯”：011，“喜欢”：010，“夺冠”：00

屏幕快照%202018-07-17%20下午6.18.43.png

**3.2 背景知识**

在这一节我来复习一下语言模型的知识，word2vec是要生成词向量，而词向量与语言模型有着密切的关系。

**3.2.1 统计语言模型**

当今的互联网迅猛发展，每天都产生大量的文本，图片，语音和视频数据。要对这些数据进行处理，从中挖掘有用的信息，离不开自然语言处理技术。在自然语言处理中，统计语言模型是很重要的部分。它被称为自然语言处理的基础，被广泛的应用于语音识别，机器翻译，分词和词性标注和信息检索等任务。

例1:在语音识别中，对给定的Voice，需要找到一个使概率最大的文本片段。我们借用Bayes公式可以得到：

在上公式中是声学模型，而是语言模型。

简单的来说统计语言模型是用来计算一个句子的概率的概率模型。它通常基于语料库来构建。那么什么是句子的概率？假设表示为一个句子由个词组成。则表示是为这句子的概率，每个词的联合概率。

利用bayes公式可以链式的分解为：

其中的条件概率，，，… ， 就是语言模型的参数。如果这些参数数据都已经计算得到，那么给一个句子就可以很快的计算出一个句子的相应概率。看起来不是很复杂，但是具体实现来说还是有点麻烦，先看看模型的参数个数。刚才考虑的是给定个词的句子，就需要计算个参数。我们假设语料库的词汇量是，即语料库中与个不同的词汇。那么如果考虑长度为的任意句子，理论上就有种可能，而每个句子又会有个参数，所以这样总共需要个参数。所以不管是时间复杂度还是空间复杂度都非常大。

这些参数如何计算得到，常见的方法有N-gram模型，决策树，最大熵模型，最大熵马尔科夫模型，条件随机场，神经网络等。我们这里先看一下N-gram模型和神经网络两种方法。

**3.2.2 N-gram模型**

考虑的近似计算，由bayes公式可以得到：

根据大数定律，当语料库足够大时可以近似的表示为：

其中和分别表示词串和在语料库中出现的次数，可以想象到当很大时和的统计计算会很耗时。

从上面的公式中可以看到一个词出现的概率与其之前的每一个都有关系，如果假定一个词的出现只会与其之前的几个词相关呢？这就是N-gram模型思想。它做了一个N-1阶的Markov假设，认为一个词的出现只与它前面的N-1个词相关，得到的公式是：

如果我们设定N=2的话，我们就会得到以下的公式：

这样来使得参数计算变得更加的容易。

那么在使用的过程中N-gram的N设定为多少来合适？一般来说N的选择要同时考虑计算复杂度和模型的效果两个因素。

模型参数量与N的关系

|  |  |
| --- | --- |
| N | 参数量 |
| 1 | 2105 |
| 2 | 21010 |
| 3 | 21015 |
| 4 | 21020 |

如果假定词汇量的大小为200000。上表大致给出了模型的参数量。事实上模型的参数量是词汇量的指数函数，显然N不能取值很大。实际应用中大部分取到3或4就够了很少取更大的。

在模型效果上，理论上N越大效果越好。现在基于大规模语料库和高性能计算机使得在N-gram模型中也可以计算N>10这样情况。但是也会知道，当N达到一定程度时，模型的效果提升就会变缓慢，提升变换变小。

在N-gram中还有比较重要的操作就是平滑化。具体的内容可以参考文献[3]。

总结一下，N-gram是语言模型的一种，其主要工作是在语料库中计算各个词串出现的次数以及平滑处理。概率值计算好之后就保存下来，等需要计算一句话的概率值时只需要找到相应的概率值连乘起来即可。

当然了在机器学习中我们通常有一种这样思路：对所需要解决的问题先建模后再为其构造一个损失函数（或目标函数）然后对这样的目标进行优化，从而得到一组最优的参数，最后用这组最优参数的模型进行预测计算。

对统计语言模型来说，利用最大似然，可以把目标函数设定为：

在上式子中表示语料库(Corpus)，表示的是词的上下文，即词周边的词的集合。当为空时，就取。对于我们之前的N-gram模型来说就有C。

通常的话对数似然用的比较多，即把目标函数改为：

然后我们对这个函数进行优化(最大化)。可见已经被设定为关于和的一个函数，即：

其中为待定参数，这样我们队上面的的目标函数一旦优化完成就会得到一个最优解，这样的话函数关系就唯一确定，以后任何的这样的函数值就会计算得到根据函数。与之前我们讨论的N-gram相比，这样的话不需要保存很多的概率统计值，直接通过计算来获得，而参数的数据量又会远远小于参数N-gram所需要的参数量。

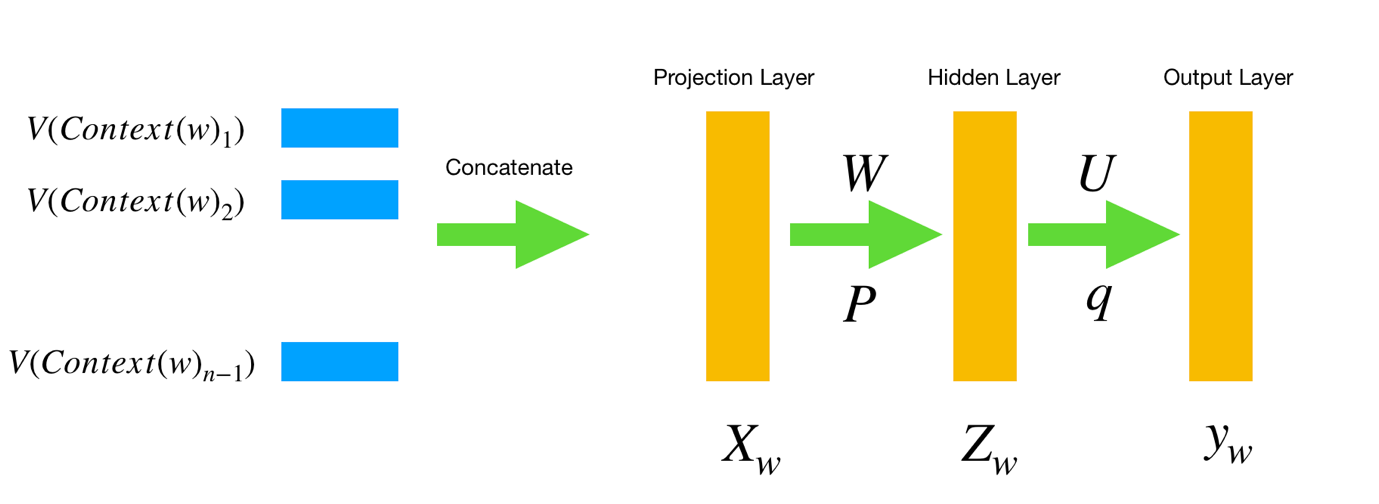
很显然，对于这样的思路，最关键的地方还是你数的构造即建模的过程。我们接下来看看如何用神经网络来构建一个这样的函数。

**3.2.3 神经概率语言模型**

本小节我们先来看看Bengio等人在论文[4]提出的神经概率语言模型。该模型用到了哟个重要的工具—词向量。

什么是词向量？简单的来说，对词典中的任意词，指定一个固定长度的实值向量就称为词的词向量，是向量的维度。

下面我们来看看其网络的结构:



对于语料库中***C***中的任意一个词，将取为前面的个词。这样模型的数据二元对就是就是一个训练样本了。一旦语料库和词向量的长度给定了，上面网络的投影层和输出层就已经确定了，前者为,后者为，即语料库词典的长度。而隐藏层的神经元个数是可以自己设定大小的。

为什么投影层的规模是？因为输入层包含中的个词的词向量，而投影出层的向量就是讲这个词的词向量首尾相接拼起来，所以其长度就是维了。之后的模型具体计算如下：

其中双曲正切函数，用来做激活函数，表示为对向量中的每一个元素同意的函数处理。经过上面两步的计算已经得到一个输出向量(假设，即语料库的词典长度为)。这样的输出向量还不能做为一个概率分布，最后做一下SoftMax归一化:

上式中的表示词在语料库词典中的索引。这样的向量就表示为在给定一个上下文后下一个词的分布。

对应到上面我们提到的，这样的函数就是这个神经网络了，而它的参数又包含在哪里呢？总的来说，有两部分：

* 词向量矩阵：，
* 神经网络参数：，，,

这些参数都需要在模型训练中得到，值得注意的是通常在机器学习中，输入都是已知的，而在上面的模型红输入也需要经过训练才能得到。

下面我们来看一下这样的模型的运算量，在网络中投影层，隐藏层，输出层的规模分别为：，，。依次来看看其中具体的参数大小。

* 是一个词的上下文词的数量，通常在5以内。
* 是一个词的词向量长度，通常是101-102数量级
* 是用户定义的隐藏层规模，通常是102数量级
* 定义为词汇量的大小，通常是104-105数量级

整个网络模型的运算都集中在隐藏层与输出层的矩阵运算，以及输出层归一化处理。

相比于，神经概率语言模型又有哪一些优势呢？主要有这两点：

1. 词语之间的相似度可以通过计算词的向量得到。

比如在语料中=“A dog is running in the room”出现了10000次。而=“A cat is running in the room”值出现了一次。按照模型肯定会远远大于。而和的区别就在于 dog和cat。而这两个词无论从语义上还是语法上都扮演着同样的角色。因此和应该很近才是。

然而，由神经语言模型中计算的和是大致上相似的，原因在于(1)在神经与语言模型中假定了“相似的”词会有相似的词向量(2)概率函数关于词向量是光滑的，即词向量中某个位置上的微小变化对最后的概率值上的变化也是微小的。因此和是大致上相似的。而这样的结果就是对于下面的句子：

A dog is running in the room

A cat is running in the room

The cat is running in the room

The cat is walking in the room

只要在语料库中出现一个，相应的其他句子的概率也会随着变大。

1. 基于词向量的语言模型自带平滑功能，不需要像N-gram 那样进行处理。

最后我们来总结想想，词向量哎整个神经语言里面模型里面扮演这怎样的角色？训练时，它是用来帮助构造目标函数的辅助参数，训练完成后，它好像又是一个语言模型的副产品，但这个副产品有时候又很有作用，接下来我们来看看其作用。

**3.2.4 词向量的理解**

在自然语言处理任务中，我们将自然语言交给机器学习算法来处理，单机器无法直接理解自然语言，首先要做的工作就是讲自然语言数字化。如何将自然语言数字化，词向量提供了很好的方式。

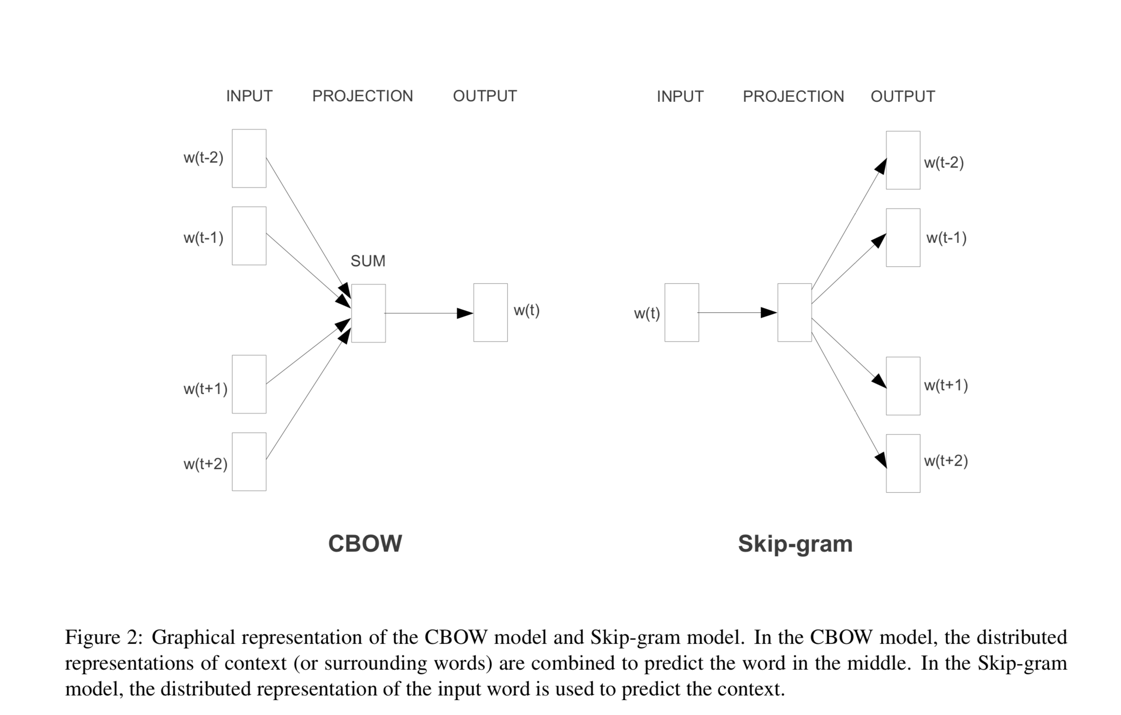
一种最简单的词向量是One-hot编码，但是这种编码有一些缺点，容易受到维度灾难的困扰，尤其将其用到深度学习算法中，它又不能很好的刻画出词与词之间的关系。

另一种词向量是Distributed Representation，最早在文献[5]中，为了克服One-hot编码的缺点，通过训练将每个词映射一个短的向量，所有的向量构成的词向量的空间，每个词又可以看做这个空间的饿一个点。在这个空间上引入距离就可以判断他们之间的相似度了。Word2Vec就是采用这样的思想。

如何获得词向量呢？有很多模型来估计词向量，包括LSA和LDA。此外，用神经网络来做也是比较常用的，比如上面的神经概率语言模型。而大部分情况下，词向量和语言模型都深捆绑在一起的，训练完成后两者是同时可以得到。一份好的词向量在自然语言处理任务中很有价值。

**3.3 基于Hierarchical Softmax的模型**

有了前面的准备，我们开始正式介绍word2vec，它包括两个模型CBOW(Continuous Bag of Words Model)和Skip-gram(Continuous Skip-gram Model)关于这两个模型，作者Tomas Mikolov在论文[6]中给出了如下示意图:



根据上图可以看到两个模型都是只有三个网络才层，输出层，投影层，输出层。前者是已知当前词的上下文，，，，的前提下来预测当前词。而后者恰恰相反是在已知当前词的情况下预测当前词的语境，，，。对于CBOW和Skip-gram两个模型，word2vec给出了两套框架，它们分别基于Hierarchical SoftMax 和 Negative Sampling来进行设计。接下来我们首先来看看基于Hierarchical SoftMax的CBOW模型和Skip-gram模型。

在之前的章节中我们有提到，基于神经网络的语言模型通常设定对数似然函数作为其目标函数：

语言模型的关键在于概率函数的构造，而之前的神经概率语言模型就是一种构造的方式。在基于Hierarchical SoftMax的CBOW模型中其优化目标也如上式所示，而Skip-gram模型中优化目标正好反过来：

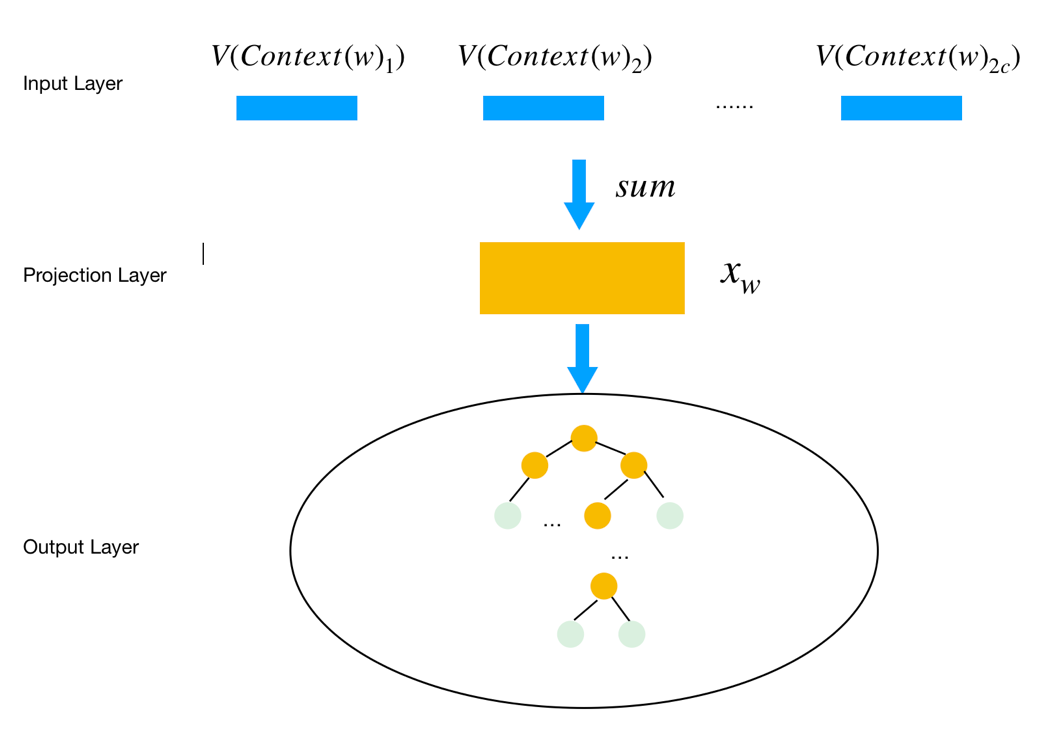
因此我们我们的主要目标是在和的构造上。接下来我们就看看这两个模型的详细介绍。

**3.3.1 CBOW模型**

CBOW模型的网络结构如下图所示，它包括三个网络层：输入层，投影层，输出层。下面以样本为例，在这里我们假设为当前词的前后各个词组成。

输入层：中的个词的词向量，… 。这里的是向量的维度。

投影层：包含的个向量做求和累加，即



输出层：输出层U对应一棵二叉树，它是将语料中出现过的词当做叶子结点，以语料中出现的次数当权重构造出来的Huffman树。在这个Huffman 树中叶子结点共有个。分别对应着词典中的每一个词。非叶子节点共有N-1个，在图中用黄色的标记出来。

对比前面的神经概率语言模型，可以知道他们有三个地方不一样：

* 从输入层到投影层的操作前者是通过拼接，后者是通过累加求和。
* 前者有隐藏层，后者没有
* 输出层，前者是线性结构，后者是树形结构。

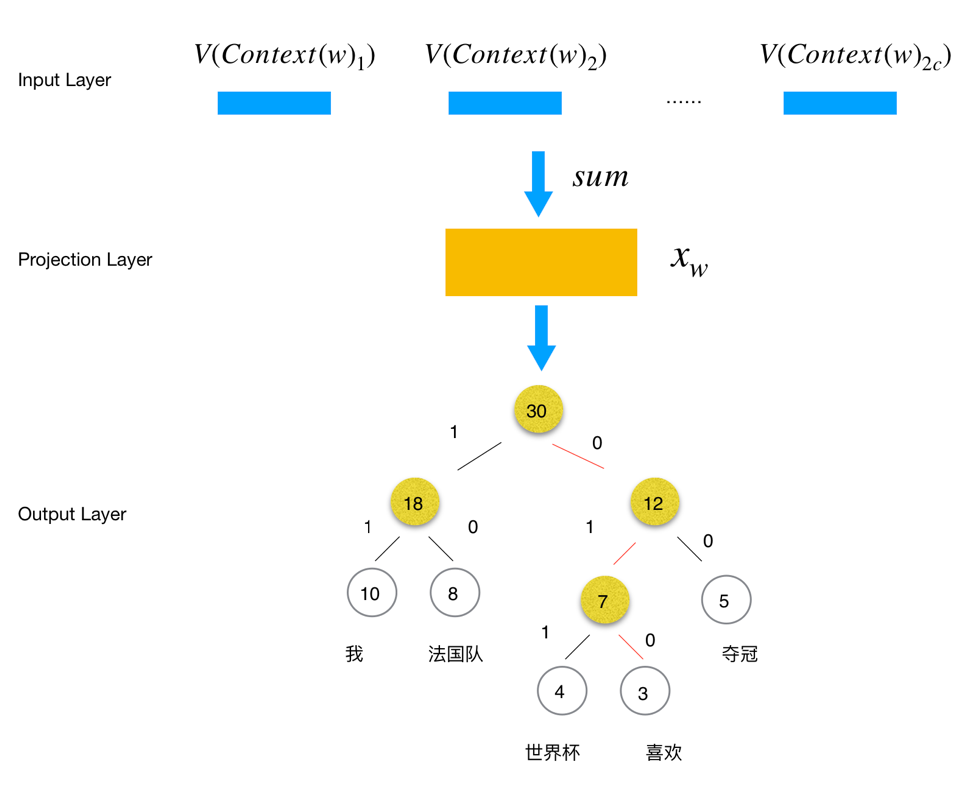
在神经概率语言模型中，我们可以看到模型的大部分计算，集中在隐藏层和输出层之间的矩阵运算，以及输出层的SoftMax运算。从上面的对比可以看到CBOW模型对计算复杂度高的地方做了相应的改变。首先去掉了隐藏层，其次输出层改用了Huffman 树，从而为Hierarchical SoftMax 技术奠定基础。

梯度计算：

Hierarchical SoftMax 是Word2vec中提高性能的关键技术，为了描述方便起见，在具体介绍这个计算之前，我们先引入若干的记号。对于Huffman树种的某一个叶子节点，假设它对应着词典中的词，记：

1. :从根结点出发到对应叶子结点的路径
2. :路径包含的结点数
3. ,,…,:路径中个结点，其中表示根结点，表示词对应的结点。
4. ,,…, :词的Huffman编码，它由位编码构成，标书路径中第个结点对应的编码，其中根结点没有对应的编码。
5. ,,…,：路径中非叶子结点对应的向量。表示路径中第个非叶子结点对应的向量。

好了有了这么多的表示约定后，我们来结合一个具体的例子看看它到底是如何实现的，还是结合上面的例子，考虑词”喜欢”的情况。



在上图由红色的线串起来的4个点构成，其长度=4.，，，为路径上的5个结点。对应的根结点，,，分别是0，1，0。即“喜欢”一词的编码是010。此外,，分别表示上三个非叶子结点表示的向量。

那么问题来了，如何用这样的网络来定义概率函数。更具体的来讲，就是如何利用向量以及Huffman树来定义函数？

以图中的词”喜欢”为例，从根结点出发，到达词这个叶子结点，一共经历了三个分支，而每一次分支我们又可以看做一次二分类问题。

既然是二分类问题，就需要对每一个分叶子结点指定一个类别，即那个是正类，那个是负类。特别巧的是，除了根结点外，树的每一个结点都对应着一个取值0或1的Huffman编码。因此，一种自然的想法是，把编码1个看做正类，把编码0的看做一个负类。我们约定：

简单的来说将一个结点进行分类时，分到左边的正类，分到右边的是负类。而有了之前我们的预备知识，可以看到一个结点被分为正类的概率是：

被分为负类的概率就是1-。注意，上面的式子中的，它是待定参数，显然在这里叶子结点对应的那些向量就是的值。

对于从根结点出发达到“喜欢”这个叶子结点所经历3次二分类，将每一个分类结果的概率写出来的话就是如下：

1. 第一次：=1-
2. 第二次：=
3. 第三次：=1-

但是我们需要的是，它跟上面的几个概率又有什么关系呢？很容易就可以看出来。

到此为止，通过关于“喜欢”的小例子，Hierarchical SoftMax的基本思想其实就已经介绍完了。

总结一下：

对于词典*D*中的任意一个词,Huffman 树中必存在一条从根结点到这个词对应的结点路径。路径上存在-1个分支，将每个分支看做一个二分问题，每次分支都产生一个概率值，将每一个概率值连乘起来就是我们需要的。

最终的条件概率值可以写为：

其中：

其实也可以将上面的饿式子写成一个式子：

最后改变我们的目标函数：

为了求导方便，将上式子求和符号中内容记为：

到现在我们已经得到CBOW的目标函数，接下来看看怎么来优化它，即如何使得它最大化。很显然我们可以用随机梯度上升法，从而求解目标函数的梯度变得很关键。接下来我们来讨论关于上面的目标函数梯度的如何求解。

于是的迭代更新公式为：

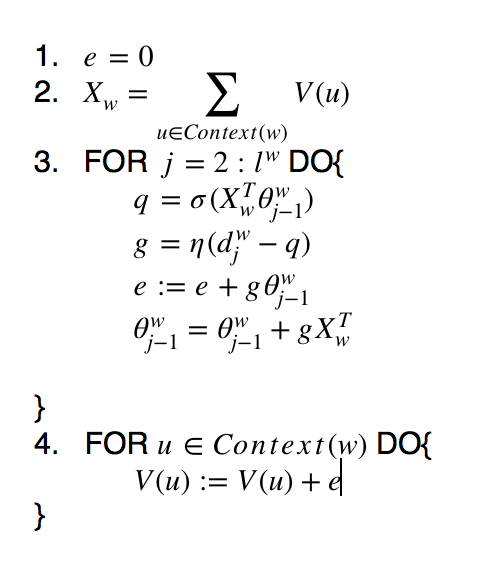
其中是学习率。

接下来考虑关于的梯度，观察上面的式子可以知道和是对称的(即可以交换位置)，因此在梯度计算上也可以将两者交换位置即可：

到这里后我们还有一个问题，我们的目标是最终获得词典D里每一个词的向量，而这里的是的累加和。那么如何用对每一个，进行更新呢？word2vec做法很简单，直接用这个梯度应用到每一个：

这样做也很好理解，既然这个来自于每个上下文的词向量之和，那么求完梯度后也应该分配到每一个分量向量上去。

现在我吗终于把这部分讨论完，最后看看这边的伪代码：



这样我们应该会回CBOW模型有了更好的理解和认识。

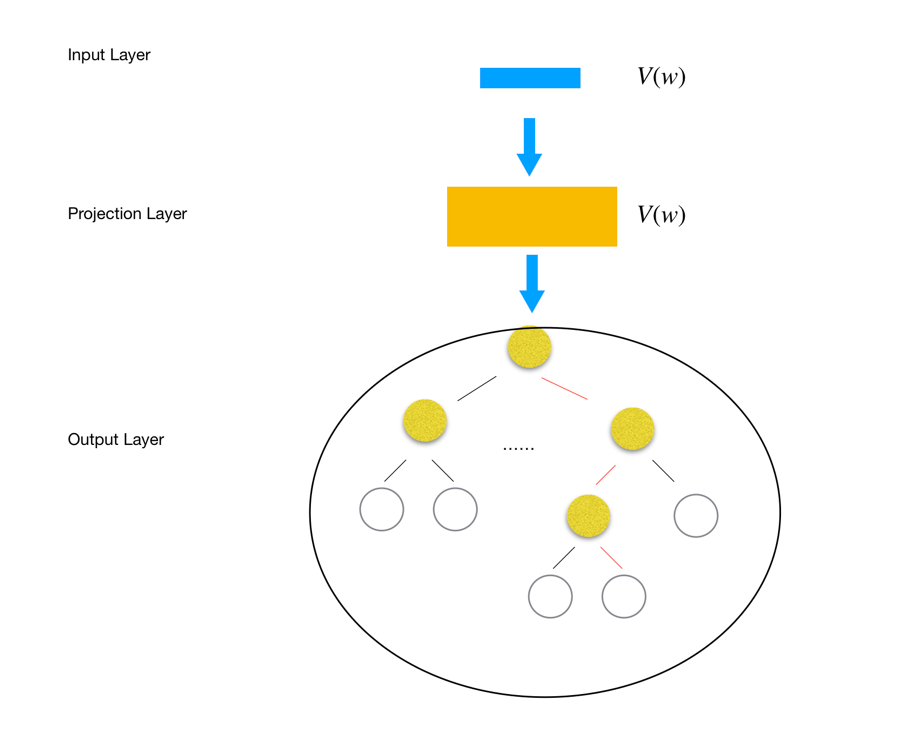
**3.3.2 Skip-gram模型**

我们接下来看看Skip-gram模型因为推到过程中很多地方都是和CBOW相似的，还继续沿用之前的符号表示，下面以样本为例，下图给出了其网络结构。

输入层：只含当前中心词的一个向量，.

投影层：这个只是恒等投影，吧复制到。所以其实是多余的，这里保留这个投影层只是为了和之前的CBOW方便做对照，。

输出层：和CBOW一样，也是输出一棵Huffman 树。



梯度计算：对于Skip-gram模型，已知的是当前词，需要对其上下文进行预测，所以目标函数还是和上面的差不多，关键的地方还是函数的构造和建模。模型中将这样的函数定义为：

上式中的上节介绍CBOW时类似的构造出来：

其中：

同样将其带回对数似然函数得到以下式子：

同样，为了求导时方便，将三重求和符里面的内容记为:

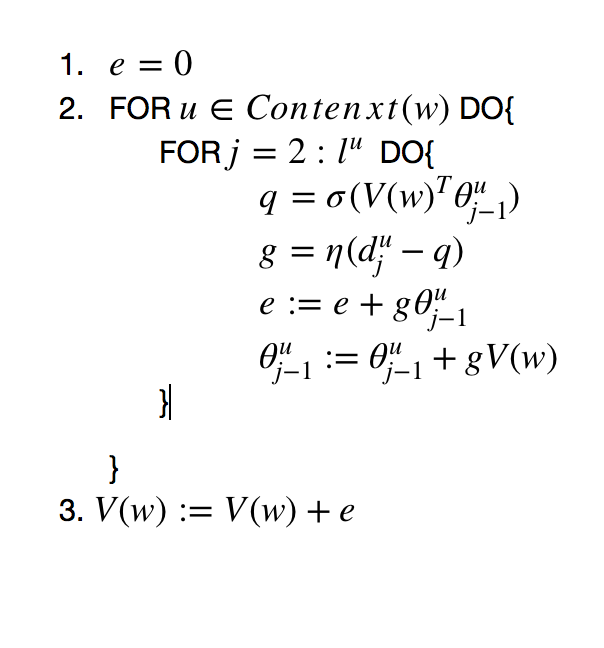
到目前为止，我们已经推倒出Skip-gram模型的目标函数，同样我们需要对其进行梯度上升去优化，现在看看具体的对求梯度过程。

于是，的更新公式可以写成：

接下来考虑关于的梯度，同样利用中和对称性，所以有：

于是的更新公式可以写成：

最后我们根据样本为例，给出Skip-gram模型中采用梯度上升法更新参数的伪代码。



好了到目前为止基于Hierarchical SoftMax的两个CBOW和Skip-gram模型都介绍完了。大家也有了透彻的理解认识。接下了我们看看基于Negative Sampling的实现。

**3.4 基于Negative Sampling 的模型**

本节我们来介绍基于Negative Sampling的CBOW和Skip-gram模型。Negative Sampling简称NEG，是Tomas Mikolov 在论文[2]中提出的，目的是提高词训练速度和改善词向量的质量。与Hierarchical SoftMax 相比NEG不在需要复杂的Huffman树，而是利用随机负采样的，可以大幅度提高性能，因而可以做Hierarchical SoftMax的替代方案。

**3.4.1 CBOW模型**

在CBOW模型中，已知词的上下文需要预测，因此给定，词就是正样本了，其他词就是负样本了。负样本那么多，我们改如何选取呢？

假定在已经选择了一个关于词的负样本子集.且对于，定义为：

表示词的标签，即正样本为1，负样本为0。对于给定一个样本对,我们希望最大化：

其中：

也可以把上面的式子写成一个式子，

这里的仍表示为中各个词的和，表示一个词所对应的词向量，为待优化的参数变量。

为什么要最大化，我们来看看表达式：

其中是用当前上下文预测当前词是的概率，是预测词时的概率。从形式上来看，最大化上面的就是最大化，同时最小化。这就是我们需要的结果，增大正样本的概率，同时减少负样本的概率。于是给定一个语料库的话，下面函数：

就可以作为优化的整体目标，当然为了计算方便，对记为：

还是为了方便推导，我们将括号内的内容记为：

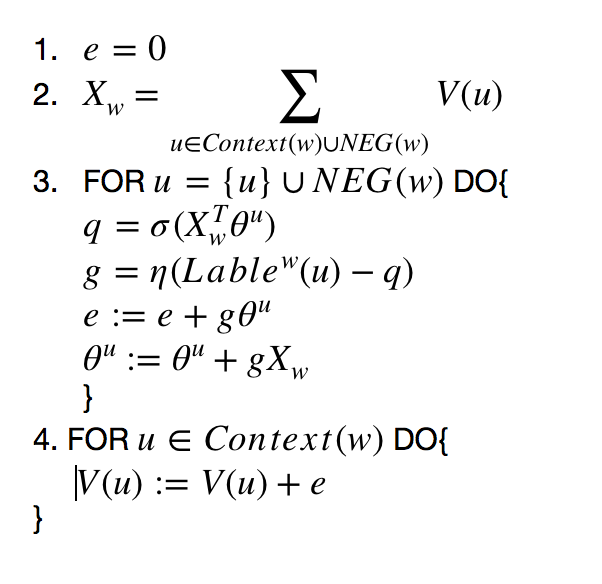
接下来利用随机梯度上升法对上面的式子优化，关键点还是计算梯度。首先还是考虑关于的梯度计算。

于是，的更新公式可以写为：

接下来还是来看看关于的梯度计算。同样是利用对称性有：

于是利用可以对，的更新公式：

下面以样本为例，给出基于Negative Sampling的CBOW模型红采用随机梯度上升法更新各参数 伪代码：



**3.4.2 Skip-gram模型**

本节介绍基于Negative Sampling的Skip-gram模型，有了之前的经验，这里我们首先将优化目标函数由原来的

改写为：

这里的表示对于给定的一个样本,我们希望最大化的量，，定义为：

其中NEG(u)表示处理词时产生的负样本子集，条件概率是：

或者写成整体的式子：

同样的操作，我们取对数，最终目标函数就是：

到这里的话我们的也只剩下计算梯度和参数更新了，和之前的操作一样。

值得注意的是在word2Vec中具体实现Negative Sampling的Skip-gram模型时并不是像上面的公式中所表示的一样。上面的公式中对于每一个样本,需要做的是对中的每一个词进行负采样。而具体的实现时我们只对进行了次负采样。

给定一个样本，我们希望最大化

其中：

或者写成一个整体：

上面的表示的是处理词时产生的负采样，于是对于一个给定的语料库*C*，函数：

就可以作为整体的优化目标，同样我们取其对数，最终的目标是：

同样为了计算导数方便起见，将括号里面的内容记为：

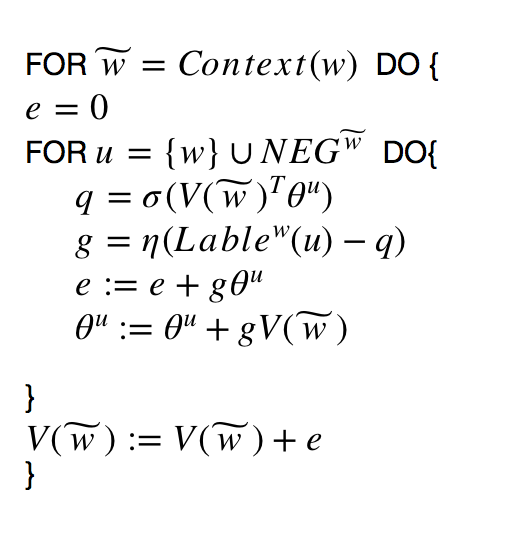
接下来利用随机梯度上升法进行优化，关键还是求出其梯度，首先考虑关于的梯度：

So,的更新公式为：

接下来我们看看关于的梯度，利用关于和的对称性有：

于是得到的更新公式：

到现在为止，关键的推导我们都差不多做完了。我们来看看具体的伪代码：



好了，现在我们也介绍完了Negative Sampling的Skip-gram模型。但是还有一个问题，负采样如何实现。

**3.4.3 负采样算法**

负采样在基于Negative Sampling的CBOW和Skip-gram模型中都很重要，那么问题来了，给定一个词,如何产生呢？

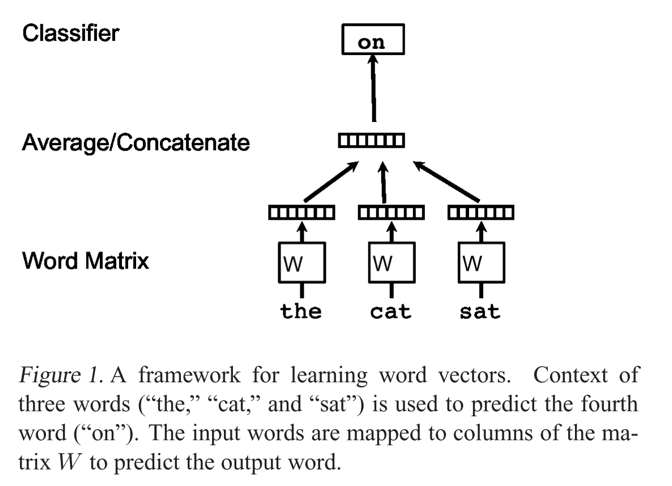
词典在语料中出现的次数有高有低，对于那些高频词，被选择的概率就比较大，反之，对于那种低频词被选择的概率就会比较小。这是我们对负采样的具体大致要求。本质也是一个带权采样过程。这里的带权采样算法有很多，就不具体介绍了，如果在采样中正好才进去了本身,那怎么办？放弃这次再来一次就行了

**3.5 句子向量**

前面我们介绍完词向量模型，总结下来就是将语料库中的一个词转化为一个实值向量，那问题来了，我们能不能把一句话转成向量呢？传统的方法有很多可以实现，tf-idf，词袋模型等。不过现在我们主要介绍一种借用词向量的句向量学习模型。论文还是Tomas Mikolov大牛所做。当然了这是2014年出来的论文，所以现在也可能会有很多新的方法，大家可以自己去搜索查找，这里就不再说了。直接我们来看论文吧，

**3.5.1词向量模型**

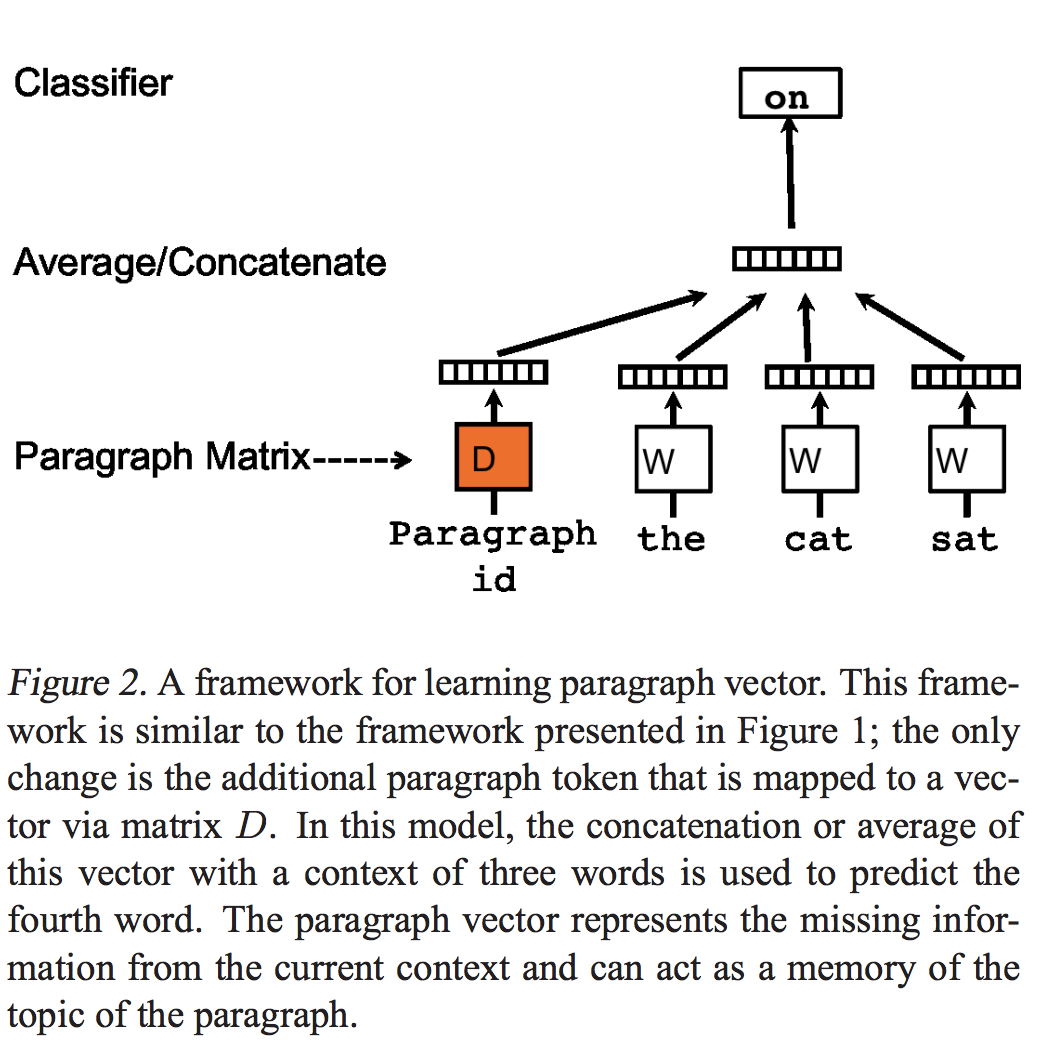
在之前的我们介绍词向量模型中，类似于CBOW框架，我们给定于一个词的语境来预测这个词。如下下图所示：



这样的模型大家已经很熟悉了，目标函数和优化过程也都可以自己推导完成，但是如何借用上面词向量的思想，我们来考虑训练一个句子向量呢？

**3.5.2句向量模型**

有一种训练句向量的思路，是把每句话定一个向量，让其参与上面的词向量的训练，思路也很简单，具体做法如下图所示：



正如论文中所描述的，这个句子向量就是提供了词向量训练中依据一个词的上下文推测这个词的额外信息，类似于这个句子的主题信息。OK，大概就这了，给大家介绍句子向量的一种实现方法，具体的推导过程留给大家参考大牛的论文去自己实现。

**3.6 总结**

词向量这部分的知识就介绍到这里了，我们来总结一下，词向量模型关键是P函数的一个建模过程，而词向量我们会作为一个待训练的参数，然后根据已经建好的模型所推导出的目标函数持续的优化，最终训练结束，就得到了训练好的参数，代表着我们的得到词向量。最后我们也了解了一下句子向量，实现过程也算是借鉴了词向量的思想，额外的给定了一个句子向量参与词向量训练过程。总之在深度学习中我们大部分是在做对问题建模的工作，也是学术界研究的主要方向。

1. Mikolov T, Chen K, Corrado G, et al. Efficient estimation of word representations in vector space[J]. arXiv preprint arXiv:1301.3781, 2013.
2. Mikolov T, Sutskever I, Chen K, et al. Distributed representations of words and phrases and their compositionality[C] Advances in neural information processing systems. 2013: 3111-3119.
3. 吴军. 数学之美[M]. 2012.
4. Bengio Y, Ducharme R, Vincent P, et al. A neural probabilistic language model[J]. Journal of machine learning research, 2003, 3(Feb): 1137-1155.
5. Rumelhart D E, Hinton G E, Williams R J. Learning representations by back-propagating errors[J]. nature, 1986, 323(6088): 533.
6. Mikolov T, Le Q V, Sutskever I. Exploiting similarities among languages for machine translation[J]. arXiv preprint arXiv:1309.4168, 2013.
7. Le Q, Mikolov T. Distributed representations of sentences and documents[C] International Conference on Machine Learning. 2014: 1188-1196.