中文词向量的演变

由于词具有广阔的丰富性，我们需要机器去代表它的含义，如何才能体现一个词的含义呢，

WordNet

它是采用相近词的方式来表达词的含义，它需要构建一系列相近词库。由此它具有以下缺点

相近词的微小差异无法表现出来

一词多义的现象也没法表达

主观性

要求大量人工去表达相近词

没法计算相近词的准确

one-hot开始，但是它是用统计的方法来的，它是一种离散的数值表现，它没有体现词语中顺序的关系，也没有一词多义的体现，词库维度太大等等各个缺点。

分布式表达

根据此前面的内容来表达词的含义，它的一种代表就是语言模型（N-gram），它也是一种统计的方法，根据贝叶斯算法公式计算概率，所以它适合推荐排序上面。同理它的缺点也来源于N-gram的缺点

数据稀疏，难免会出现OOV的问题

随着n的增大，参数空间呈指数增大

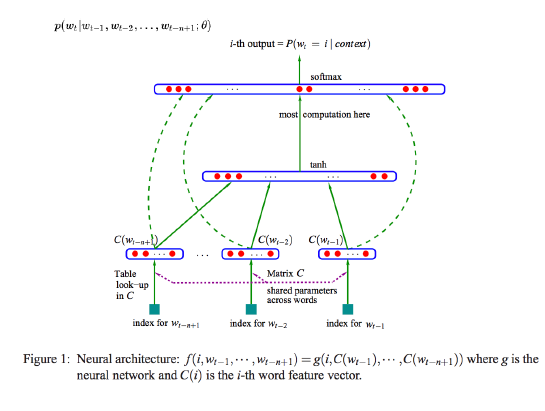
缺少长期依赖，只能建模到前n-1个词

一词多义

Word vector

它需要将所有的词投射到同一个向量空间里面，并且其中近义词距离相邻的思想。

NPLM



模型

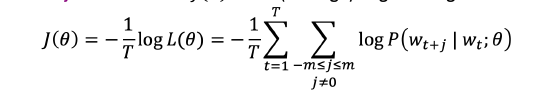
它的思想是输入前文单词来预测下文，但是这种效果不佳，它的中间产物引出了词向量

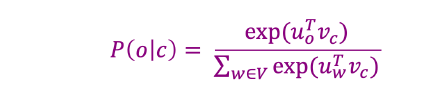
它的缺点在于有限的前文信息，计算量过大

Word2vec

它的基本想法非常大的预料集，每一个单词都能表达一个向量，文本中的每一个位置都有一个center word和context word,我们要用相似度计算使得词的上下文与词的向量很近，我们最大化这种概率

它的目标函数

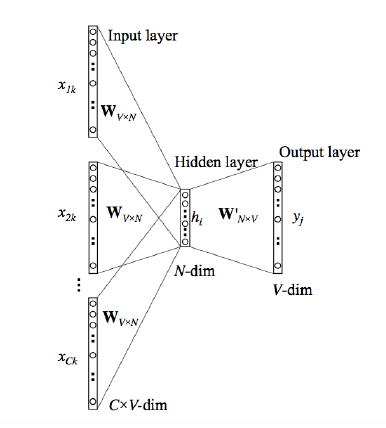




我们使用点积来计算o与c的相似性

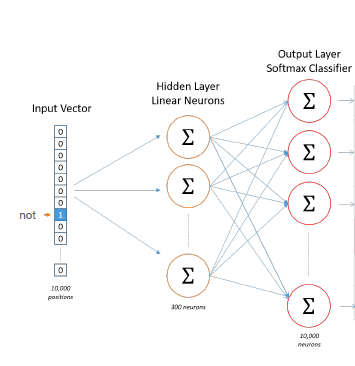
两种方式

CBOW



它是通过两边词来预测中间值

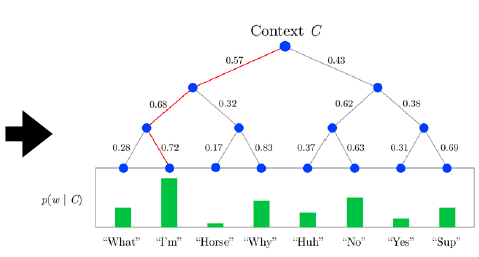
Skip-gram



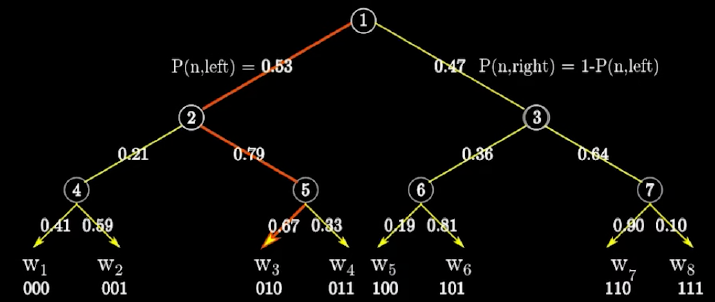
它是通过中间词来预测周边单词。当我们训练好模型，获得中间的词向量矩阵，然后将输入值的one-hot与之相乘得到的就是输入值的词向量。

这两种方式的不同点cbow相对简单，训练速度也就相对快的。CBow能更好学习同义的关系。Skip-gram学习的是语义的关系

缺点都有计算复杂。所以引入了H-softmax（分层）

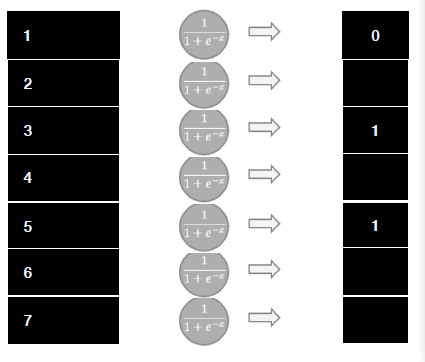


它是将最后的softmax转换成一个二叉树的方式，从而将复杂度从O(n)到O(logn)。先判断它是经过哪个节点（判断的方法采用lr方法）然后逐步分层，叶子节点就是单词。对于单词节点采用编码的方式



概率的来源我们将输出的神经元对应树里面的每一个节点， 对应每一个节点都采用sigmoid

然后构建下面模型



将上文输入预测下文，将上文输入进去通过向量矩阵得到每个节点，每个节点后面跟着一个sigmoid函数。而预测的单词对应的编码，是我们的节点最终要实现的，这是一种判定决策的过程。

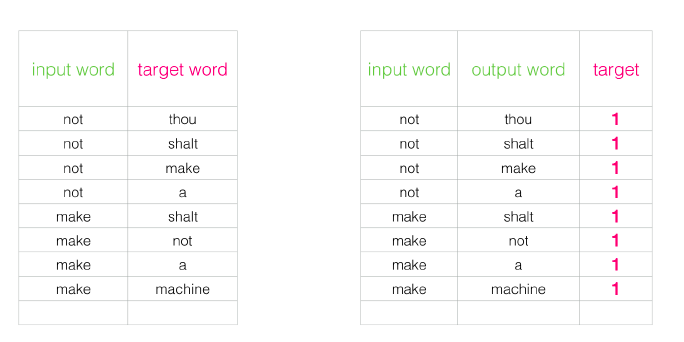
Huffman的构建可以采用单词的频次，这就造成了频次低的单词经过的路径就会比较长，计算量同意复杂。

负采样

Softmax正常版本是先embedding,然后计算分，最后通过softmax预测值的。

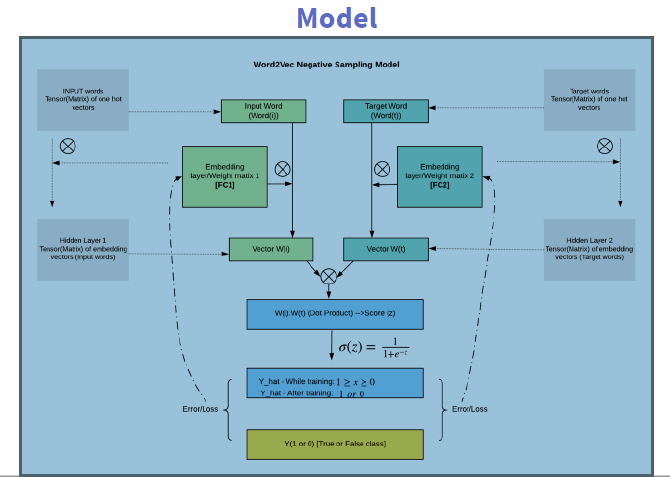
现在我们转换成LR任务，原先通过上下文预测词，现在是将上下文和预测值一起输入LR来预测概率，概率过大则是判断是正样本。

原先skip-gram的方式转换就成了下面



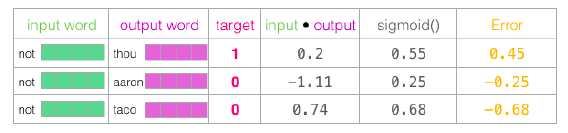
其中target代表着是否是正样本

如果按照原先的skip构建，则全是正样本，学习不到很好的信息。所以我们需要随机构建来增加不是相邻词构成负样本。



模型

整个模型的训练过程类似



从上面的模型图，整个过程我们可以看到先将input和output输入训练模型，然后将其与模型中的两个初始化embedding矩阵相乘，然后将input和output相乘，在通过sigmoid函数得到分数，将target的值与分数相减，得到Error,然后将error反向传播，反复训练的过程。

整个模型参数一个是window size 和负采样数目。