Word2vector

# One-hot编码

One-hot编码就是把每个样本中的单个特征用一个向量表示，该向量只有1个维度是1，其他的都是0。

One-hot的编码过程不需要学习和训练。但是One-hot编码的向量是高维度且极其稀疏的，大量的位置为0，计算效率较低，同时也不利于神经网络的训练。从语义角度来讲，One-hot 编码还有一个严重的问题，它忽略了单词先天具有的语义相关性。比如，对于单词“like”、“dislike”、“Rome”、“Paris”来说，“like”和“dislike”在语义角度就强相关，它们都表示喜欢的程度；“Rome”和“Paris”同样也是强相关，他们都表示欧洲的两个地点。对于一组这样的单词来说，如果采用One-hot编码，得到的向量之间没有相关性，不能很好地体现原有文字的语义相关度。

其缺点是：（1）向量的维度会随着句子的词的数量类型增大而增大；如果将世界所有城市名称对应的向 量合为一个矩阵的话，那这个矩阵过于稀疏，并且会造成维度灾难。（2）编码是随机的，向量之间相互独立，无法表示语义层面上词汇之间的相关信息。

所以，提出了对独热编码的改进方法：（1）将vector每一个元素由整形改为浮点型，变为整个实数范围的表示；（2）转化为低维度的连续值，也就是稠密向量。将原来稀疏的巨大维度压缩嵌入到一个更小维度的空间。并且其中意思相近的词将被映射到向量空间中相近的位置。

简单说，要寻找一个空间映射，把高维词向量嵌入到一个低维空间。然后就可以继续处理。

# 分布式词向量编码

分布式编码（Distributed representation）最大的贡献就是让相关或者相似的词，在距离上更接近了。向量的距离可以用传统的欧氏距离来衡量，也可以用余弦相似度来衡量。

将word映 射 到 一 个 新 的 空 间 中 ， 并 以 多 维 的 连 续 实 数 向 量 进 行 表 示 叫 做 “Word Represention” 或 “Word Embedding”。自从21世纪以来，人们逐渐从原始的词向量稀疏表示法过渡到现在的低维空间中的密集表示。用稀疏表示法在解决实际问题时经常会遇到维数灾难，并且语义信息无法表示，无法揭示word之间的潜在联系。而采用低维空间表示法，不但解决了维数灾难问题，并且挖掘了word之间的关联属性，从而提高了向量语义上的准确度。

# 词向量模型类型

## LSA矩阵分解模型

采用线性代数中的奇异值分解方法，选取前几个比较大的奇异值所对应的特征向量将原矩阵映射到 低维空间中，从而达到词矢量的目的。

## PLSA潜在语义分析概率模型

从概率学的角度重新审视了矩阵分解模型，并得到一个从统计，概率角度上推导出来的和LSA相当 的词矢量模型。

## LDA文档生成模型

按照文档生成的过程，使用贝叶斯估计统计学方法，将文档用多个主题来表示。LDA不只解决了同义词的问题，还解决了一次多义的问题。目前训练LDA模型的方法有原始论文中的基于EM和差分贝叶斯方法以及后来出现的Gibbs Samplings 采样算法。

## Word2Vector模型

近几年刚刚火起来的算法，通过神经网络机器学习算法来训练N-gram语言模型，并在训练过程中求出word所对应的vector的方法。

# Word2Vector

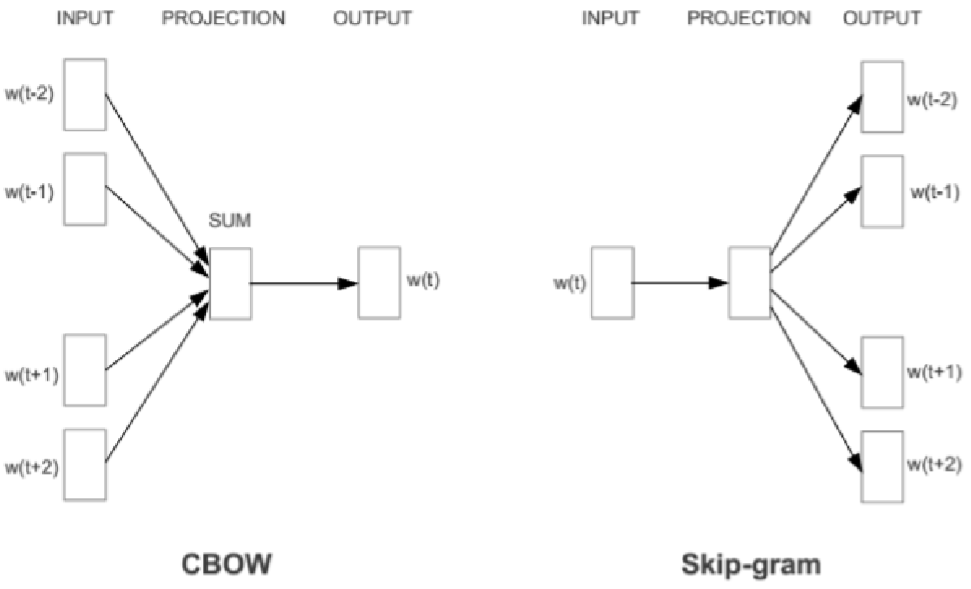
word2vec就是将词表征为实数值向量的一种高效的算法模型，其利用深度学习的思想，可以通过训练，把对文本内容的处理简化为K维向量空间中的向量运算，而向量空间上的相似度可以用来表示文本语义上的相似。

Word2vec输出的词向量可以被用来做很多NLP相关的工作，比如聚类、找同义词、词性分析等。如果换个思路，把词当做特征，那么Word2vec就可以把特征映射到K维向量空间，可以为文本数据寻求更加深层次的特征表示 。

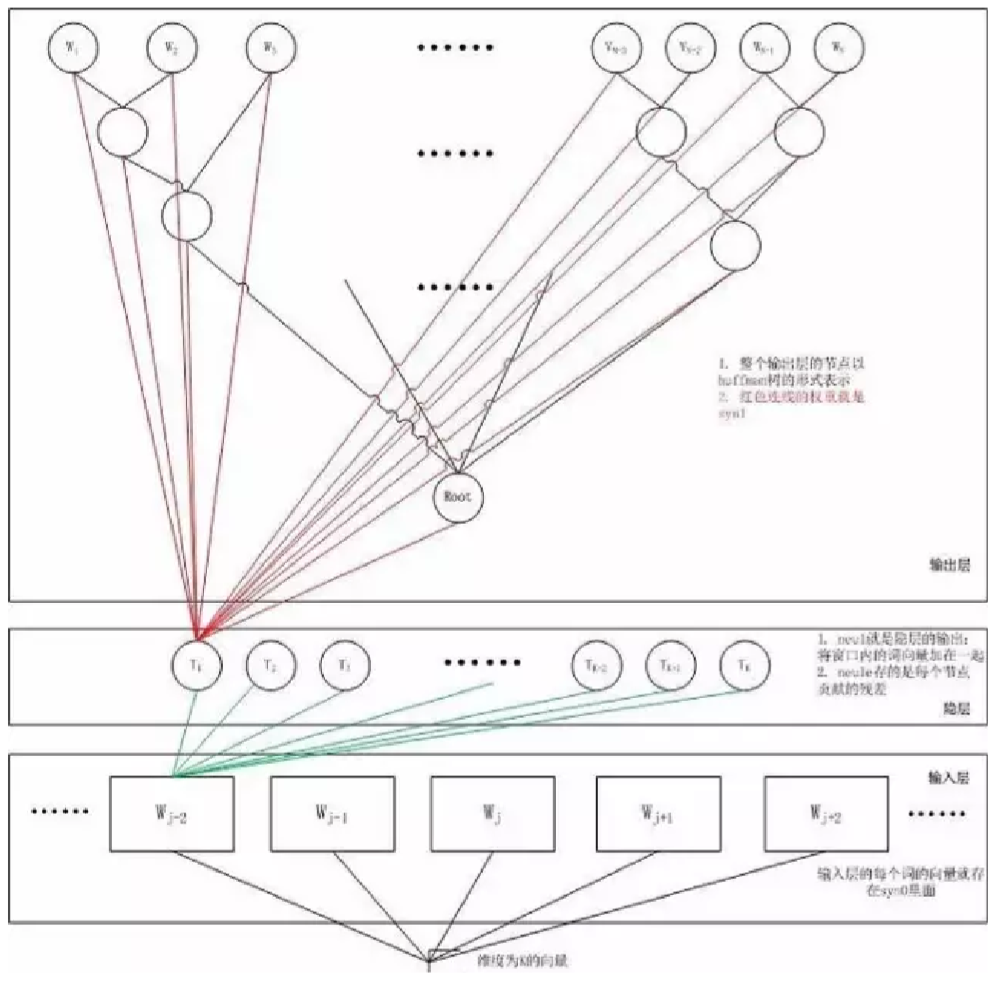
Word2vec属于分布式表示。其基本思想是：通过训练将每个词映射成K维实数向量（K一般为模型中的超参数），通过词之间的距离（比如 cosine 相似度、欧氏距离等）来判断它们之间的语义相似度。其采用一个三层的神经网络，输入层-隐层-输出层。有个核心的技 术是根据词频用Huffman编码，使得所有词频相似的词隐藏层激活的内容基本一致，出现频率越高的词语，他们激活的隐藏层数目越少，这样有效的降低了计算的复杂度。而Word2vec大受欢迎的一个原因正是其高效性，，一个优化的单机版本一天可训练上千亿词。

与潜在语义分析（Latent SemanticIndex Analysis, LSA）、潜在狄立克雷分配（Latent Dirichlet Allocation，LDA）的经典过程相比，Word2vec利用了词的上下文，语义信息更加地丰富。

Word2Vec有两种不同的方法：Continuous Bagof Words (CBOW)和Skip-gram。CBOW的目标是根据上下文来预测当前词语的概率。Skip-gram刚好相反：根据当前词语来预测上下文的概率（如下图所示）。这两种方法都利用人工神经网络作为它们的分类算法。起初，每个单词都是一个随机N维向量。经过训练之后，该算法利用CBOW或者Skip-gram的方法获得了每个单词的优向量。



取一个适当大小的窗口当做语境，输入层读入窗口内的词，将它们的向量（K维，初始随机）加和在一起，形成隐藏层K个节点。输出层是一个巨大的二叉树，叶节点代表语料里所有的词（语料含有V个独立的词，则二叉树有|V|个叶节点）。而这整颗二叉树构建的算法就是Huffman树。这样，对于叶节点的每一个词，就会有一个全局唯一的编码，形如"010011"，不妨记左子树为1，右子树为0。接下来，隐层的每一个节点都会跟二叉树的内节点有连边，于是对于二叉树的每一个内节点都会有K条连边，每条边上也会有权值。



对于语料库中的某个词w\_t，对应着二叉树的某个叶子节点，因此它必然有一个二进制编码，如"010011"。在训练阶段，当给定上下文，要预测后面的词w\_t的时候，我们就从二叉树的根节点开始遍历，这里的目标就是预测这个词的二进制编号的每一位。即对于给定的上下文，我们的目标 是使得预测词的二进制编码概率大。形象地说，我们希望在根节点，词向量和与根节点相连经过logistic计算得到bit=1的概率尽量接近0，在第二层，希望其 bit=1 的概率尽量接近1，这么一 直下去，我们把一路上计算得到的概率相乘，即得到目标词w\_t在当前网络下的概率P(w\_t)，那么对于当前这个sample的残差就是1-P(w\_t)，于是就可以使用梯度下降法训练这个网络得到所有的参数值了。显而易见，按照目标词的二进制编码计算到后的概率值就是归一化的。

Hierarchical Softmax用Huffman编码构造二叉树，其实借助了分类问题中，使用一连串二分类近似多分类的思想。例如我们是把所有的词都作为输出，那么“桔子”、“汽车”都是混在一起。给定w\_t 的上下文，先让模型判断w\_t是不是名词，再判断是不是食物名，再判断是不是水果，再判断是不是“桔子”。

在训练过程中，模型会赋予这些抽象的中间结点一个合适的向量，这个向量代表了它对应的所有子结点。因为真正的单词公用了这些抽象结点的向量，所以Hierarchical Softmax方法和原始问 题并不是等价的，但是这种近似并不会显著带来性能上的损失同时又使得模型的求解规模显著上升。

没有使用这种二叉树，而是直接从隐层直接计算每一个输出的概率——即传统的Softmax，就需要对|V|中的每一个词都算一遍，这个过程时间复杂度是O(|V|)的。而使用了二叉树（如Word2vec中 的Huffman树），其时间复杂度就降到了O(log2(|V|))，速度大大地加快了。

现在这些词向量已经捕捉到上下文的信息。我们可以利用基本代数公式来发现单词之间的关系（比 如，“国王”-“男人”+“女人”=“王后”）。这些词向量可以代替词袋用来预测未知数据的情感状况。该模型的优点在于不仅考虑了语境信息还压缩了数据规模（通常情况下，词汇量规模大约在300个单词左右而不是之前模型的100000个单词）。因为神经网络可以替我们提取出这些特征的信息，所以我们仅需要做很少的手动工作。但是由于文本的长度各异，我们可能需要利用所有词向量的平均值作为分类算法的输入值，从而对整个文本文档进行分类处理。

参考：

龙书，TensorFlow深度学习

<https://mp.weixin.qq.com/s/XZkQpAWZqFJWfqyIXlMmfg>

<https://mp.weixin.qq.com/s/un6e1KYhKlNJ-rGruazoXw>