H2 Word2vec总结

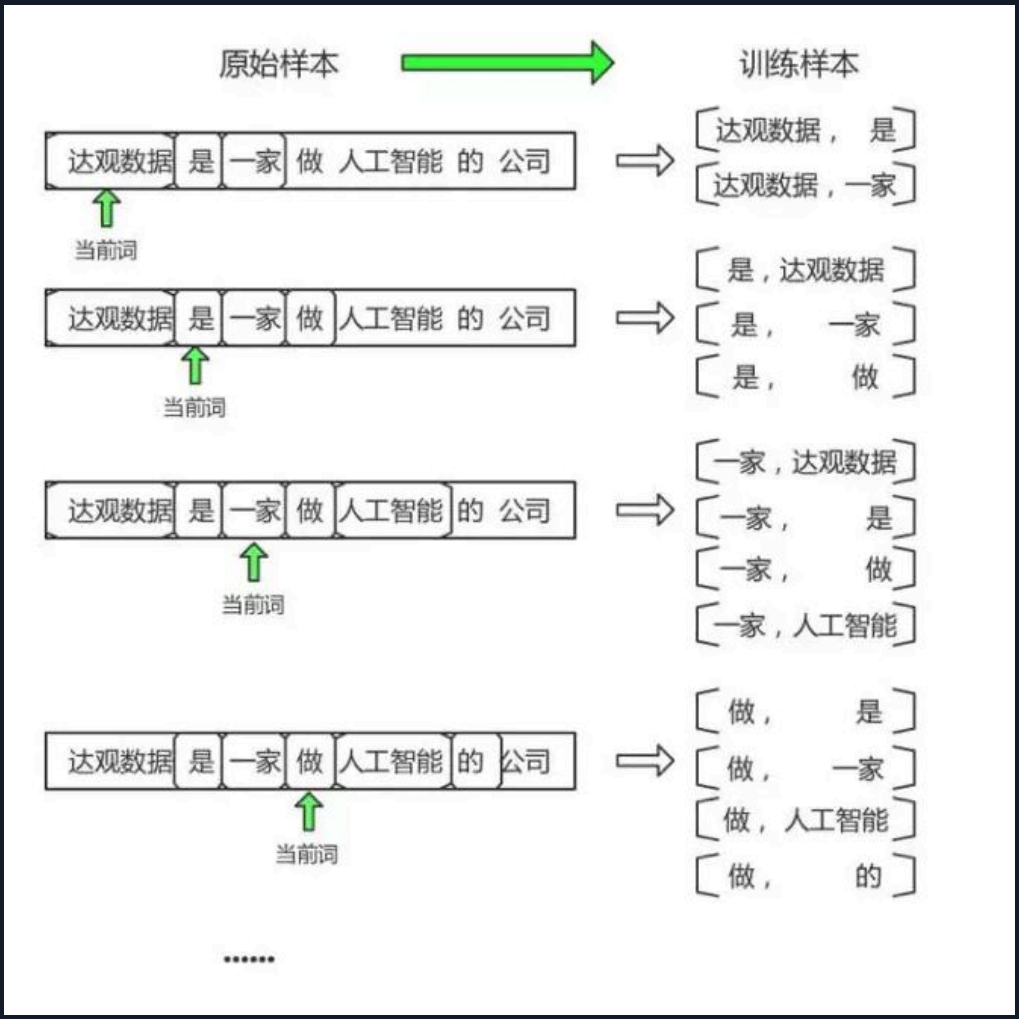
H3 一、常用模型结构

H4 1.1 ship-gram模型

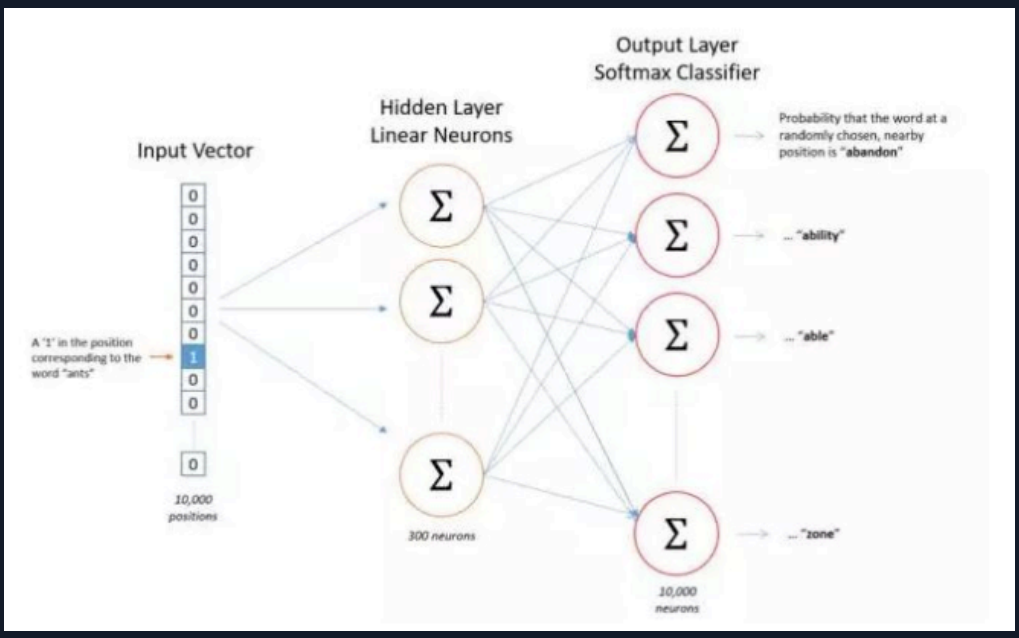
> 根据中心词来预测周围词；

H5 1.1.1 训练样本

> skip-gram模型的输入是当前词，输出是当前词的上下文，虽然我们训练模型的时候喂的是一个个分词好的句子，但内部其实是使用一个个word pair来训练。样例如下图：



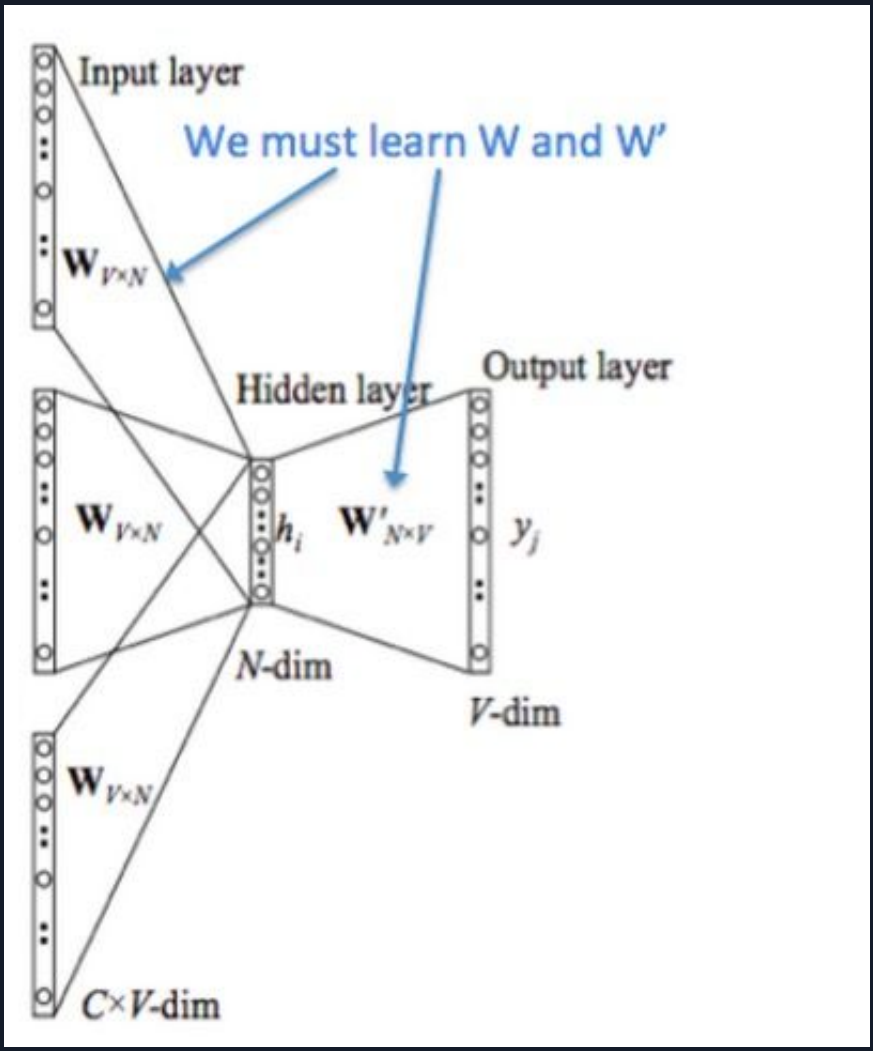
H5 1.1.2 模型结构



H4 1.2 连续词袋模型（CBOW）

> 根据中心词周围的词来预测中心词

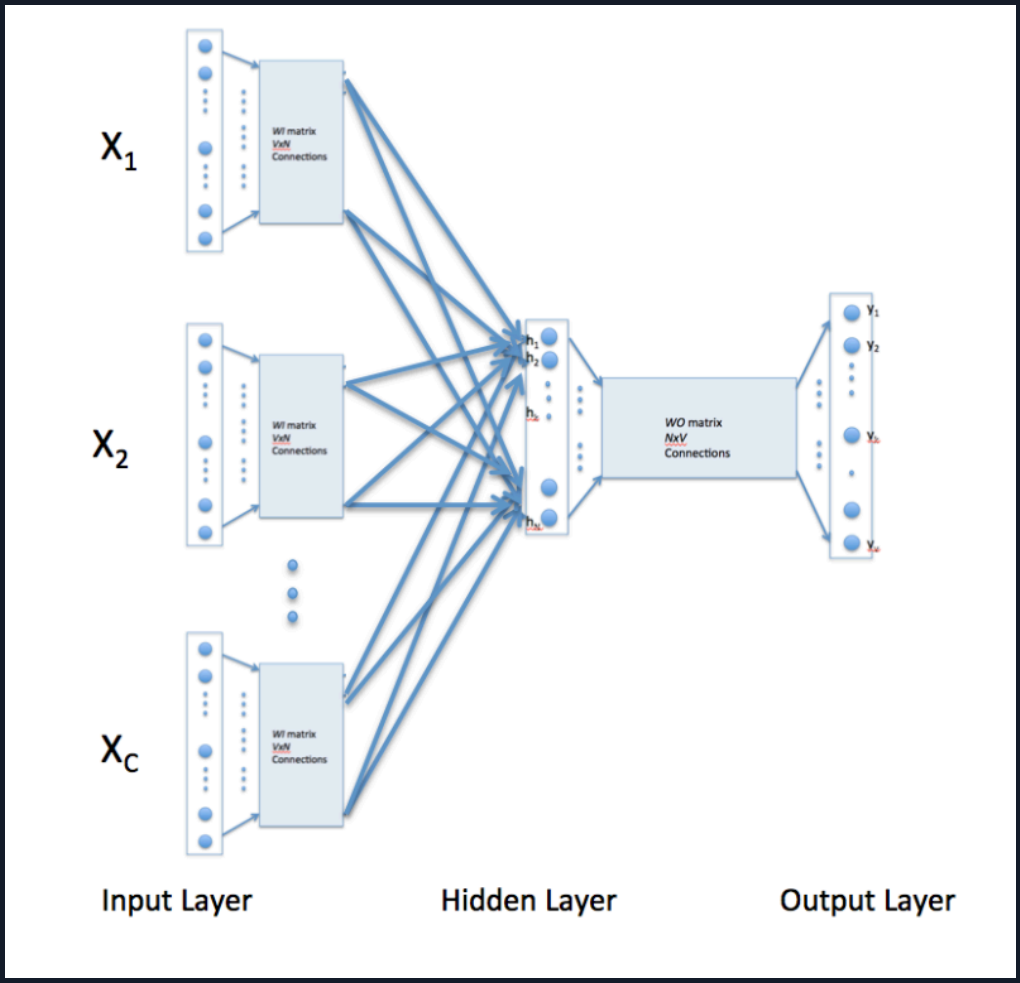
H5 1.2.1 模型结构



假设单词的向量空间维度为V，上下文单词个数为C，求解两个权重均值W和W'。对于上图的解释如下：

- 输入层:上下文单词的onehot形式；
- 隐藏层:将输入层所有onehot后的向量乘以第一个权重矩阵W（所有的权重矩阵相同，即共享权重矩阵），然后相加求平均作为隐藏层向量，该向量的大小与输入层的每一个样本大小相同；
- 输出层:将隐藏层向量乘以第二权重矩阵W‘，得到一个V维的向量，然后再通过激活函数（softmax）得到每一维词的概率分布，概率最大的位置所指示的单词为预测出的中间词；
- 一般使用使用的损失函数为交叉熵损失函数，采用梯度下降的方式来更新W和W'；这实际上是一个假任务，即我们需要的只是第一个权重矩阵W。得到第一个矩阵W之后，我们就能得到每个单词的词向量了。

H5 1.2.2 详细流程



H4 1.2.3 训练样例

> 假设我们现在的语料是这一个简单的只有四个单词的句子：
> I drink coffee everyday
> 我们使用的window size设为2。
>
> 则训练这么一个句子，我们会需要训练4个batch，即句子中的单词个数的batch。
>
> - 首先对单词onehot则可以得到
>
> X_I = [1,0,0,0]；X_drink = [0,1,0,0]；X_coffee = [0,0,1,0] ； X_everyday = [0,0,0,1]；
>
> - 具体步骤
>
> 1. 第一个batch：I为中心词，drink coffee为上下文，即使用单词drink coffee来预测单词I，即输入为X_drink和X_coffee，输出为X_I，然后训练上述网络；
> 2. 第二个batch：drink为中心词，I和 coffee everyday为上下文，即使用单词I和coffee everyday，即输入为X_I和X_coffee、X_everyday，输出为X_drink，然后训练上述网络；
> 3. 第三个batch：coffee为中心词，I coffee 和 everyday为上下文，同理训练网络；
> 4. 第四个batch：everyday为中心词，drink coffee为上下文，同理训练网络。
>
> - 然后重复上述过程（迭代）3-5次（epoehs）左右即得到最后的结果。

H3 二、二次采样

> 用最简单的一句话描述二次采样就是，对文本中的每个单词会有一定概率删除掉，这个概率是和词频有关，越高频的词越有概率被删掉

H4 2.1 目的

> - 提升训练速度
> - 使得单词表示的学习也更加规范
> - 有助于抵消语料库中罕见词和频繁词之间的不平衡

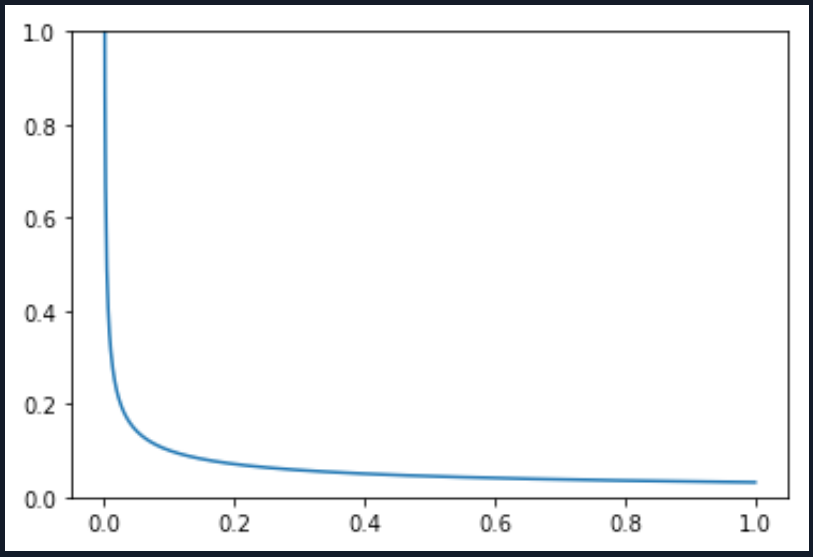
H4 2.2 公式

$$P(x) = \left(\sqrt{\frac{z(x_i)}{s} + 1} \right) * \frac{s}{z(x_i)}$$

> 其中xi是相关单词，z(xi)是训练数据集或语料库中单词的词频与总词数之比，而s是二次采样率（超参数，首选值为0.001）
>
> 二次采样率是决定是否保留频繁词的关键因素，值越小意味着单词被保留在模型训练语料库中的可能性越小。

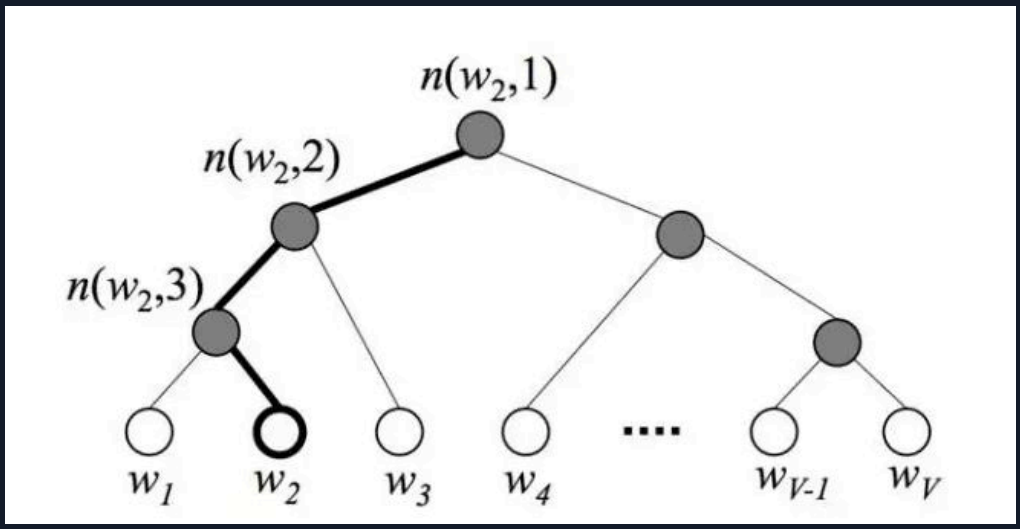
H4 2.3 图形化二次采样函数

```python
# coding: utf-8
# 频繁词二次采样
# 目的：降低计算复杂度
import numpy as np
import matplotlib.pylab as plt

def subsample(x):
    return np.sqrt(x/0.001 + 1) * (0.001/x)

X = np.arange(0, 1, 0.001)
Y = subsample(X)
plt.plot(X, Y)
plt.ylim(0, 1)  # 指定图中绘制的y轴的范围
plt.show()
```



H3 三、层次softmax

> 层次softmax的目的和负采样一样，也是为了加快训练速度，但它相对复杂，没有负采样这种来的简单粗暴。具体来说，**使用层次softmax时图4中的模型输出层不再是使用one-hot加softmax回归，而是使用Huffman树加softmax回归**。在模型训练的时候首先统计语料中词语的词频，然后根据词频来构建Huffman树，如下图所示，树的根节点可理解为输入词的词向量，叶子节点表示词表中的词，其它节点没有什么实际含义，仅起到辅助作用。



H4 3.1 加速原因

> **输出层不使用one-hot来表示，softmax回归就不需要对那么多0（也即负样本）进行拟合**，仅仅只需要拟合输出值在Huffman树中的一条路径。假设词表大小为N，一条路径上节点的个数可以用来估计，就是说只需要拟合次，这给计算量带来了指数级的减少。此外，由于Huffman编码是不等长编码，频率越高的词越接近根节点，这也使计算量有所降低。

> 怎么对树中的节点进行拟合呢？如图7所示，假设训练样本的输出词是 w_2 ，则从根节点走到 w_2 经过了 $n(w_2,2), n(w_3,3)$ 这两个节点。由于Huffman树是二叉树，这意味着只需要判断向左还是向右就可以从根节点走到 w_2 ，判断向左还是向右其实就是进行二分类。图7中的例子，“root(input)->left->left->right()”这条路径的概率可表示为：

$$P(w_2|input) = P(left|input) \cdot P(left|input) \cdot P(right|input)$$
$$= logistic(\theta_1 \cdot X_{input}) \cdot logistic(\theta_2 \cdot X_{input}) \cdot (1 - logistic(\theta_3 \cdot X_{input}))$$

$$logistic(x) = \frac{1}{1+e^{-x}}$$

> 其中 θ_i 表示路径中第i个节点的权值向量。注意一点，softmax regression 做二分类的时候就退化为了logistic regression，因此虽然叫层次softmax但公式中其实用的是logistic function。根据上述公式就可构建根据Huffman树来进行softmax回归的cost function，进而根据梯度下降对模型进行训练求解。

H3 四、负采样（negative sample）

> - 负采样是噪声对比估计（NCE）方法的简化形式，因为它在选择噪声样本（即负样本）计数及其分布时做出了假设。
> - 负采样被用作分层softmax函数的替代
> - 尽管负采样可以在训练模型时使用，但是在推断时，仍然计算完整的softmax值，以获得归一化的概率分布

> 以ship-gram所示的模型为例，对每一个训练样本需要更新的参数个数有三百万（准确的说是三百万零三百，由于输入是one-hot，隐藏层每次只需要更新输入词语的词向量），这还是假设词表只有一万的情况下，实际情况会有五十万甚至更多，这时候参数就达到了亿级。训练过程中要对每个参数计算偏导，然后进行更新，这需要很大的计算资源。
>
> **负采样是加快训练速度的一种方法**，这里的负可以理解为负样本。针对训练样本（ants, able），able这个词是正样本，词表中除able外的所有词都是负样本。负采样是对负样本进行采样，不进行负采样时，对每一个训练样本模型需要拟合一个正样本和九千九百九十九个负样本。加入负采样后，只需要从这九千九百九十九个负样本中挑出来几个进行拟合，大大节省了计算资源。那么应该挑几个负样本，根据什么进行挑呢？Google给出的建议是挑**5-20个**，怎么挑是根据词在语料中出现的概率，概率越大越有可能被选中。

H4 4.1 负采样概率公式

$$P(word) = \frac{f(word)^{\frac{3}{4}}}{\sum_{i=1}^{10000} f(word_i)^{\frac{3}{4}}}$$

> 其中f(word)标识word出现的概率

H3 五、应用场景

H4 5.1 特征降维

> 特征维度过高的时候，很容易出现特征之间具有较高的相关性。这种情况下可以利用词向量工具对特征进行聚类，将相关的特征归到一个维度里面。

H4 5.2 特征扩展

> 针对短文本处理时，一个case往往提不出很多表意较强的特征，导致类别间区分度不强。这种情况下可以利用词向量工具对主要特征进行扩展，在不损失精度的前提下提高召回。

> 《面向自然语言处理的深度学习（用Python创建神经网络）》
>
> 2020年了，你竟然还不知道word2vec的二次采样
>
> 漫谈Word2vec之skip-gram模型
>
> 究竟什么是Word2vec？Skip-Gram模型和Continuous Bag of Words(CBOW)模型？
>
> Word2vec之CBOW
>
> word2vec是如何得到词向量的？