文本向量化

到目前为止，我整理了NLP的分词和关键词提取。接下来我们想如果把一段文本经过分词之后，该怎么进一步通过数学的方式 去描述这些词？我们的机器学习领域该如何处理这些词？显然基于神经网络的自然语言处理模型来说直接处理这些词是不合理 的。这个时候将文本向量化就被提出来了。起初词袋的向量模型被发明出来，但是人们发现词袋模型有很多弊端，比如：

- 向量的维度受词典大小控制，容易产生稀疏矩阵，高维度向量运算有很大的性能瓶颈。
- 词袋模型没有考虑词语之间的顺序关系，对语义理解来讲是一个极大的信息浪费
- 由于所有向量之间都是正交关系，无法表达相似度，会造成语义鸿沟现象

基于词袋模型有很多弊端，然后词向量就被发明出来。那什么是词向量呢？简单的说就是用一个一定维度的向量来表示词典里 的词。经过对词典里面的词进行训练，得到词向量，得到这个词向量之后NLP威力就大增，比如：

- 可以实现同义词匹配
- 通过多维向量表示，也能更为方便的进行计算。例如，“女人”+“漂亮” + “皮肤白” + “有钱” =“白富美”

那么问题来了，既然词向量那么厉害，我们该如何得到词向量呢？ Google的大神在2013年发表了一篇论文，论文中把word2vec正式的提了出来。word2vec是一套基于神经网络的词嵌入工具， 主要包含2个模型，skip-gram模型和CBOW模型。接下来我就这两个模型的学习做了一些笔记供自己日后参考，也打算写2个例子 来帮助自己加深理解其中的数学过程。

在说明这两种模型之前我把一些概念名词做一些说明

- 中心词，目标词：选定的词
- 上下文，背景词：被选定词的左右两边的词
- 窗口大小：取上下文的词数 比如我有这么一个经过分词之后词组： 【人 如果 没用 梦想 跟 咸鱼 还有 什么 差别】 如果我窗口大小假设为2，那么如果我选【梦想】这个词为中心词的时候，那么它的上下文就【如果 没用 跟 咸鱼】

另外在讨论数学算法之前把softmax 函数的概念也补充一些： 在多分类问题中，我们经常会使用softmax函数作为网络输出层的激活函数 softmax函数可以回输入值进行归一化处理，把所有的输入值转化为概率， 所以的概率值加起来等于1， softmax公式为： $softmax(x)*{i}=|frac{exp(x*{i})}{\sum_j exp(x_{j})}$

例如某个神经网络有3个输出值为： 【1， 5， 3】 那经过有softmax公式可以计算出 p1=$\frac{e^1}{e^1+e^5+e^3}$ = 0.016 p2= $\frac{e^5}{e^1+e^5+e^3}$ = 0.867 p3=$\frac{e^3}{e^1+e^5+e^3}$ = 0.117 所以经过softmax函数之后数值就变成了【0.016， 0.867， 0.117】

skip-gram

有人把它翻译为【跳字模型】 简单的来说就是基于某个词来生成它在文本序列周围的词。也就是说我给定一个词的时候，以最大概率生成 上下文。用数学来表示就是说，我给定中心词【梦想】时，生成与它距离不超过2个词的背景词【如果 没用 跟 咸鱼】 的条件概率，即：

$$P(\text{" 如果 "}, \text{" 没用 "}, \text{" 跟 "}, \text{" 咸鱼 "} \mid \text{" 梦想 "})$$

这个时候我们有一个重要假设：背景词的生成时相互独立的，那么就大大简化了条件概率的计算：

$$P(\text{" 如果 "} \mid \text{" 梦想 "}) \cdot P(\text{" 没用 "} \mid \text{" 梦想 "}) \cdot P(\text{" 跟 "} \mid \text{" 梦想 "}) \cdot P(\text{" 咸鱼 "} \mid \text{" 梦想 "}) \cdot$$

接下来是skip-gram的数学描述： 在skip-gram模型中，每个词都被表示成2个d维向量，用来计算条件概率。假设这个词在 词典中的索引为i，当它为中心词是向量表示为$v_i \in \mathbb{R}^d$，而作为背景词时 向量表示为$u_i \in \mathbb{R}^d$。假设中心词w_c在词典中索引为c，背景词w_o在词典中的索引为o，给定中心词生成背景词的条件概率可以通过对向量的点积然后做softmax计算得到：

$$P(w_o|w_c) = \frac{exp(u_o^\top v_c)}{\sum_{i\in\mathcal{V}} exp(u_i^\top v_c)} \cdots\cdots\cdots\cdots\cdots\cdots ①$$

其中词典索引集$\mathcal{V} = \{0, 1, 2, \ldots, |\mathcal{V}| - 1\}$

假设给定一个一个长度为T的文本序列，设时间步长t的词为$w^{(t)}$。我们同样假定背景词的生成时相互独立的，当背景窗口大小为m时，它的似然函数（中心词生成所有背景词的概率）为:

$$\prod_{i=1}^{T} \prod_{-m\le j\le m, j\neq 0} P(w^{(t+j)}|w^{(t)})$$

训练skip-gram模型

该模型的参数时每个词所对应的中心词向量和背景词向量。我们通过最大化似然函数来学习 模型参数，也就是人们常说的极大似然估计。也就是等价于最小化以下的损失函数：

$$-\sum_{t=1}^{T} \sum_{-m\le j\le m, j\neq 0} \log P(w^{(t+j)}|w^{(t)})$$

如果我们通过梯度下降法，那么在每一次迭代里面我们随机采样一个比较短的子序列来计算 有关该子序列的损失，然后计算梯度来更新模型参数。 根据公式①的定义我们可以得到：

$$\log P(w_o|w_c) = u_o^\top v_c - \log\left(\sum_{i\in\mathcal{V}} exp(u_i^\top v_c)\right)$$

通过对v_c求导我们得到梯度公式：

$$\frac{\partial \log P(w_o|w_c)}{\partial v_c} = u_o - \frac{\sum_{j\in\mathcal{V}} exp(u_j^\top v_c) u_j}{\sum_{i\in\mathcal{V}} exp(u_i^\top v_c)} = u_o - \sum_{j\in\mathcal{V}} \left(\frac{exp(u_j^\top v_c)}{\sum_{i\in\mathcal{V}} exp(u_i^\top v_c)}\right) u_j = u_o - \sum_{j\in\mathcal{V}} P(w_j|w_c) u_j$$

从这个推导结果来看梯度计算是需要词典中所有词以w_c为中心的条件概率。以此类推，其他 词向量的梯度计算也是如此

训练结束之后，对应词典中的任一索引为i的词，我们都得到该词作为中心词和背景词的两组 词向量v_i和u_i。在NLP中一般使用skip-gram模型的中心词向量作为词的特征向量