机器学习算法系列（27）：word2vec（词向量与统计语言模型）

Word2Vec是Google在2013年年中开源的一款将词表征为实数值向量的高效工具，采用的模型有CBOW（Continuous Bag-Of-Words，即连续的词袋模型）和Skip-Gram两种。

word2vec一般被外界认为是一个Deep Learning（深度学习）的模型，究其原因，可能和word2vec的作者Tomas Mikolov的Deep Learning背景以及word2vec是一种神经网络模型相关，但我们谨慎认为该模型层次较浅，严格来说还不能算是深层模型。当然如果word2vec上层再套一层与具体应用相关的输出层，比如Softmax，此时更像是一个深层模型。

word2vec通过训练，可以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度。因此，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、词性分析等等。而word2vec被人广为传颂的地方是其向量的加法组合运算（Additive Compositionality），官网上的例子是：
$vector('Paris') - vector('France') + vector('Italy') \approx vector('Rome')$,
$vector('king') - vector('man') + vector('women') \approx vector('queen')$。但我们认为这个多少有点被过度炒作了，很多其他降维或主题模型在一定程度也能达到类似效果，而且word2vec也只是少量的例子完美符合这种加减法操作，并不是所有的case都满足。

word2vec大受欢迎的另一个原因是其高效性，Mikolov在论文中指出一个优化的单机版本一天可训练上千亿个词。

一、词向量

自然语言处理相关任务中，要将自然语言处理交给机器学习中的算法来处理，通常需要首先将语言数学化，因为机器不是人，机器只认得数学符合。向量是人把自然界的东西抽象出来交给机器处理的东西，基本上可以说向量是人对机器输入的主要方式了。

词向量就是用来将语言中的词进行数学化的一种方式，顾名思义，词向量就是把一个词表示成一个向量。主要有两种表示方式：

1.1 One-Hot Representation

一种最简单的词向量方式是One-Hot Representation，就是用一个很长的向量来表示一个词，向量的长度为词典的大小，向量的分量只有一个1，其他都为0，1的位置对应该词在词典中的位置。举个例子：“话筒”表示为[0001000000000000]“麦克风”表示[0000000010000000]。每一个

词都是茫茫0海中的一个1.
这种One-Hot Representation如果采用稀疏方式存储，会非常地简洁，也就是给每个词分配一个数字ID。比如刚才的例子中，话筒为3，麦克风即为8（假设从0开始记）。如果要编程的话，用Hash表给每个词分配一个编号就可以了。这么简洁的表示方法配合上最大熵、SVM、CRF等等算法已经很好地完成了NLP领域的各种主流任务了。
但这种表示方法有两个缺点：（1）容易受维数灾难的困扰，尤其是将其用于Deep Learning的一些算法时（2）不能很好地刻画词与词之间的相似性：任意两个词之间都是孤立的。光从这两个向量中看不出两个词是否有关系，哪怕是话筒和麦克风这样的同义词也不能幸免于难。

1.2 Distributed Representation

另一种就是DistributedRepresentation 这种表示，它最早是 Hinton 于 1986 年提出的，可以克服 one-hot representation 的缺点。其基本想法是直接用一个普通的向量表示一个词，这种向量一般长成这个样子：[0.792, −0.177, −0.107, 0.109, −0.542, ...]，也就是普通的向量表示形式。维度以 50 维和 100 维比较常见。

当然一个词怎么表示成这么样的一个向量是要经过一番训练的，训练方法较多，word2vec是其中一种，在后面会提到，这里先说它的意义。还要注意的是每个词在不同的语料库和不同的训练方法下，得到的词向量可能是不一样的。

词向量一般维数不高，很少有人闲着没事训练的时候定义一个10000维以上的维数，所以用起来维数灾难的机会现对于one-hot representation表示就大大减少了。

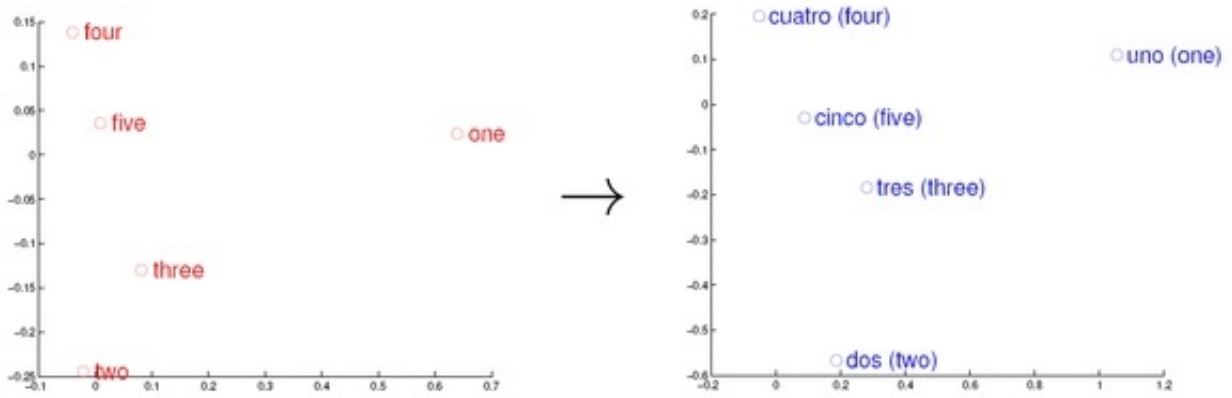
由于是用向量表示，而且用较好的训练算法得到的词向量的向量一般是有空间上的意义的，也就是说，将所有这些向量放在一起形成一个词向量空间，而每一向量则为该空间中的一个点，在这个空间上的词向量之间的距离度量也可以表示对应的两个词之间的“距离”。所谓两个词之间的“距离”，就是这两个词之间的语法，语义之间的相似性。

一个比较爽的应用方法是，得到词向量后，假如对于某个词A，想找出这个词最相似的词，这个场景对人来说都不轻松，毕竟比较主观，但是对于建立好词向量后的情况，对计算机来说，只要拿这个词的词向量跟其他词的词向量一一计算欧式距离或者cos距离，得到距离最小的那个词，就是它最相似的。

这样的特性使得词向量很有意义，自然就会吸引比较多的人去研究，前有Bengio发表在JMLR上的论文《A Neural Probabilistic Language Model》，又有Hinton的层次化Log-Bilinear模型，还有google的TomasMikolov 团队搞的word2vec等等。

词向量在机器翻译领域的一个应用，就是google的TomasMikolov 团队开发了一种词典和术语表的自动生成技术，该技术通过向量空间，把一种语言转变成另一种语言，实验中对英语和西班牙语间的翻译准确率高达90%。

介绍算法原理的时候举了一个例子：考虑英语和西班牙语两种语言，通过训练分别得到它们对应的词向量空间 E 和 S。从英语中取出五个词 one，two，three，four，five，设其在 E 中对应的词向量分别为 v1，v2，v3，v4，v5，为方便作图，利用主成分分析（PCA）降维，得到相应的二维向量 u1，u2，u3，u4，u5，在二维平面上将这五个点描出来，如下图左图所示。类似地，在西班牙语中取出（与 one，two，three，four，five 对应的） uno，dos，tres，cuatro，cinco，设其在 S 中对应的词向量分别为 s1，s2，s3，s4，s5，用 PCA 降维后的二维向量分别为 t1，t2，t3，t4，t5，将它们在二维平面上描出来（可能还需作适当的旋转），如下图右图所示：



观察左、右两幅图，容易发现：五个词在两个向量空间中的相对位置差不多，这说明两种语言对应向量空间结构之间具有相似性，从而进一步说明了在词向量空间利用刻画词之间相似性的合理性。

二、语言模型

2.1 基本概念

语言模型其实就是看一句话是不是正常人说出来的。这玩意很有用，比如机器翻译、语音识别得到若干候选之后，可以利用语言模型挑一个尽量靠谱的结果。在 NLP 的其它任务里也都能用到。
语言模型形式化的描述就是给定一个T个词的字符串s，看他是自然语言的概率$P(w_1, w_2, \cdots w_t)$，w_1到w_t依次表示这句话中的各个词。有个很简单的推论是：

$$P(s) = P(w_1, w_2, \cdots w_T) = P(w_1)P(w_2|w_1)P(w_3|w_1, w_2) \cdots P(w_t|w_1, w_2, \cdots, w_{t-1})$$

上面这个概率表示的意义是：第一个词确定后，看后面的词在前面的词出现的情况下出现的概率。如一句话“大家喜欢吃苹果”，总共四个词“大家”、“喜欢”、“吃”
、“苹果”，怎么分词现在不讨论，总之词已经分好，就这四个。那么这句话是一个自然语言的概率是：

$$P(\text{大家，喜欢，吃，苹果}) = P(\text{大家})P(\text{喜欢}|\text{大家})P(\text{吃}|\text{大家，喜欢})P(\text{苹}$$

其中，$P(大\ 家)$表示“大家”这个词在语料库里面出现的概率；$P(喜\ 欢 \mid 大\ 家)$表示“喜欢”这个词出现在“大家喜欢”后面的概率；$P(吃 \mid 大\ 家，\ 喜\ 欢)$表示“吃”这个词出现在“大家喜欢”后面的概率；$P(苹\ 果 \mid 大\ 家，\ 喜\ 欢，\ 吃)$表示“苹果”这个词出现在“大家喜欢吃”后面的概率。把这些概率连乘起来，得到的就是这句话平时出现的概率。如果这个概率特别低，说明这句话不常出现，那么就不算是一句自然语言，因为在语料库中很少出现。如果出现的概率高，就说明是一句自然语言。

看到了上面的计算，看有多麻烦：只有四个词的一句话，需要计算的是$P(大\ 家)$，
$P(喜\ 欢 \mid 大\ 家)$，$P(吃 \mid 大\ 家，\ 喜\ 欢)$，$P(苹\ 果 \mid 大\ 家，\ 喜\ 欢，\ 吃)$这四个概率，这四个概率还要预先计算好，考虑词的数量，成千上万个，再考虑组合数，
$P(吃 \mid 大\ 家，\ 喜\ 欢)$这个有“大家”、“喜欢”、“吃”的组合，总会有上亿种情况吧；再考虑$P(苹\ 果 \mid 大\ 家, 喜\ 欢，\ 吃)$这个概率，总共也会超过万亿种。

从上面的情况来看，计算起来是非常麻烦的，一般都用偷懒的方式。
为了表示简单，上面的公式可以用下面的方式表示

$$P(s) = P(w_1, w_2, \cdots w_t) = \prod_{i=1}^{T} P(w_i \mid context_i)$$

其中，如果context是空的话，就是它自己$P(w)$，另外如“吃”的context就是“大家”、“喜欢”，其余的对号入座。

2.2 上下文无关模型

该模型仅仅考虑当前词本身的概率，不考虑该词所对应的上下文环境。这是
一种最简单，易于实现，但没有多大实际应用价值的统计语言模型。

$$p(w_t \mid Context) = p(w_t) = \frac{N_{w_t}}{N}$$

这个模型不考虑任何上下文信息，仅仅依赖于训练文本中的词频统计。它是 n-gram 模型中当 n=1 的特殊情形，所以有时也称作 Unigram Model(一元文法统 计模型)。实际应用中，常被应用到一些商用语音识别系统中。

2.3 N-gram模型

接下来说怎么计算$P(w_i \mid context_i)$，上面看的是根据这句话前面的所有词来计算，那么就得计算很多了，比如就得把语料库里面的组合“大家”、“喜欢”、“吃”、“苹果”这种情况全部统计一遍，那么为了计算这句话的概率，就上面那个例子，都得扫描四次语料库。这样一句话有多少个词就得

扫描多少趟，语料库一般都比较大，越大的语料库越能提供准确的判断。这样的计算速度在真正使用的时候是万万不可接受的，线上扫描一篇文章是不是一推乱七八糟的没有序列的文字都得扫描很久，这样的应用根本没人考虑。

最好的办法就是直接把所有的$P(w_i|context_i)$提前算好了，那么根据排列组上面的来算，对于一个只有四个词的语料库，总共就有$4!+3!+2!+1!$个情况要计算，那就是24个情况要计算；换成1000个词的语料库，就是$\sum_{i=1}^{1000} i!$个情况需要统计，对于计算机来说，计算这些东西简直是开玩笑。

19世纪到20世纪初，俄罗斯有个数学家叫马尔科夫（Andrey Markov），他给了个偷懒但还颇为有效的方法，也就是每当遇见这种情况时，就假设任意一个词w_i出现的概率只同他前面的词w_{t-1}有关，于是问题就变得很简单了。这种假设在数学上称为马尔科夫假设。现在，s出现的概率就变得简单了：

$$P(s) = P(w_i)P(w_2|w_1)\cdot\cdot\cdot P(w_i|w_{i-1})\cdot\cdot\cdot P(w_n|w_{n-1})$$

这个公式对应的统计语言模型是二元模型（Bigram Model）。

接下来的问题就是如何估计条件概率$P(w_i|w_{i-1})$。根据它的定义：

$$P(w_i|w_{i-1}) = \frac{P(w_{i-1}, w_i)}{P(w_{i-1})}$$

。估计联合概率$P(w_{i-1}, w_i)$和边缘概率$P(w_{i-1})$，现在变得很简单。因为有了大量机读文件，也就是专业人士讲的语料库，只要数一数w_{i-1}, w_i这对词在统计的文本中前后相邻出现了多少次$count(w_{i-1}, w_i)$，以及w_{i-1}本身在同样的文本中出现了多少次$count(w_{i-1})$然后用两个数分别除以语料库的大小#，即可得到这些词或二元组的相对频率：

$$f(w_{i-1}, w_i) = \frac{count(w_{i-1}, w_i)}{count}$$

$$f(w_{i-1}) = \frac{count(w_{i-1})}{count}$$

根据大数定理，只要统计量足够，相对频率就等于概率，即

$$P(w_{i-1}, w_i) \approx \frac{count(w_{i-1}, w_i)}{count}$$

而$P(w_i|w_{i-1})$就是这两个数的比值，再考虑到上面的两个概率有相同的分母，可以约掉，因此

$$P(w_i|w_{i-1}) \approx \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

当然，也可以假设一个词由前面N-1个词决定，对应的模型稍微复杂些，被称为N元模型。而N=1的一元模型实际上是一个上下文无关的模型，也就是假定当前词出现的概率与前面的词无关。而在实际应用中最多的是N=3的三元模型，更高阶的模型就很少使用了。

为什么N一般取值都这么小呢？这里面主要有两个原因。首先，N元模型的大小（或者说空间复杂度）几乎是N的指数函数，即$O(|V|^N)$，这里$|V|$是一种语言词典的词汇量，一般在几万到几十万个。而使用N元模型的速度（或者说时间复杂度）也几乎是一个指数函数，即$O(|V|^{N-1})$。因此，N不能太大。当N从1到2，再从2到3时，模型的效果上升显著。而当模型从3到4时，效果的提升就不是很显著了，而资源的耗费增加却非常快。

所以N-gram存在几个问题，总结如下：

1. n-gram 语言模型无法建模更远的关系，语料的不足使得无法训练更高阶的 语言模型。大部分研究或工作都是使用 Trigram，就算使用高阶的模型，其统计 到的概率可信度就大打折扣，还有一些比较小的问题采用 Bigram。
2. 这种模型无法建模出词之间的相似度，有时候两个具有某种相似性的词， 如果一个词经常出现在某段词之后，那么也许另一个词出现在这段词后面的概率 也比较大。比如“白色的汽车”经常出现，那完全可以认为“白色的轿车”也可 能经常出现。
3. 有些n元（n个词的组合，跟顺序有关的）在语料库里面没有出现过，对应出来的条件概率就是0，这样一整句话的概率都是0了，这是不对的，解决的方法主要有两种：
    - 平滑法：最简单的方法是把每个 n 元组的出现次数加 1，那么原来 出现 k 次的某个 n 元组就会记为 k+1 次，原来出现 0 次的 n 元组就会记为出现 1 次。这种也称为 Laplace 平滑。当然还有很多更复杂的其他平滑方法，其本质都 是将模型变为贝叶斯模型，通过引入先验分布打破似然一统天下的局面。而引入 先验方法的不同也就产生了很多不同的平滑方法。
    - 回退法：即利用n-1的元组的概率去替代n元组的概率，有点像决策树中的后剪枝方法，即如果 n 元的概率不到， 那就往上回退一步，用 n-1 元的概率乘上一个权重来模拟。

2.3 N-pos模型

严格来说 n-pos 只是 n-gram 的一种衍生模型。n-gram 模型假定第 t 个词出现 概率条件依赖它前 N-1 个词，而现实中很多词出现的概率是条件依赖于它前面词 的语法功能的。n-pos 模型就是基于这种假设的模型，它将词按照其语法功能进 行分类，由这些词类决定下一个词出现的概率。这样的词类称为词性 (Part-of-Speech，简称为 POS)。n-pos 模型中的每个词的条件概率表示为:

$$p(s) = p(w_1^T) = p(w_1, w_2, \ldots, w_T) = \prod_{t=1}^{T} p(w_t \mid c(w_{t-n+1}), c(w_{t-n+2}), \ldots, c(w_{t-1}))$$

c 为类别映射函数，即把 T 个词映射到 K 个类别(1=<K<=T)。实际上 n-Pos使用了一种聚类的思想，使得原来 n-gram 中$w_{t-n+1}, w_{t-n+2}, \ldots, w_{t-1}$中的可能为$T^{N-1}$ 减少到

$c(w_{t-n+1}), c(w_{t-n+2}), \ldots, c(w_{t-1})$中的 K^{N-1}，同时这种减少还采用了语义有意义的类别。

2.4 基于决策树的语言模型

上面提到的上下文无关语言模型、N-gram语言模型、N-pos语言模型等等，都可以以统计决策树的形式表示出来。而统计决策树中每个结点的决策规则是一个上下文相关的问题。这些问题可以是“前一个词是w吗”“前一个词属于类别c_i吗”，当然基于决策树的语言模型还可以灵活一点，可以是一些“前一个是动词？”、“后面有介词吗”之类的复杂语法语义问题。

基于决策树的语言模型优点是：分布数不是预先固定好的，而是根据训练语料库中的实际情况确定，更为灵活。缺点是：构造通即决策树的问题很困难，且时空开销很大。

2.5 最大熵模型

最大熵原理是E.T. Jayness于上世纪50年代提出的，其基本思想是:对一个 随机事件的概率分布进行预测时，在满足全部已知的条件下对未知的情况不做任 何主观假设。从信息论的角度来说就是:在只掌握关于未知分布的部分知识时， 应当选取符合这些知识但又能使得熵最大的概率分布。

$$P(w \mid context) = \frac{\sum_i \lambda_i f_i(context, w)}{z(context)}$$

其中λ_i是参数，$Z(Context)$为归一化因子，因为采用的是这种 Softmax 形式， 所以最大熵模型有时候也称为指数模型。

2.6 自适应语言模型

前面的模型概率分布都是预先从训练语料库中估算好的，属于静态语言模型。 而自适应语言模型类似是 Online Learning 的过程，即根据少量新数据动态调整模 型，属于动态模型。在自然语言中，经常出现这样现象:某些在文本中通常很少 出现的词，在某一局部文本中突然大量地出现。能够根据词在局部文本中出现的 情况动态地调整语言模型中的概率分布数据的语言模型成为动态、自适应或者基 于缓存的语言模型。通常的做法是将静态模型与动态模型通过参数融合到一起， 这种混合模型可以有效地避免数据稀疏的问题。

还有一种主题相关的自适应语言模型，直观的例子为:专门针对体育相关内 容训练一个语言模型，同时保留所有语料训练的整体语言模型，当新来的数据属 于体育类别时，其应该使用的模型就是体育相关主题模型和整体语言模型相融合 的混合模型。