词向量与Embedding究竟是怎么回事？

3 Dec By 苏剑林 | 2016-12-03 | 159856位读者 引用

词向量，英文名叫Word Embedding，按照字面意思，应该是词嵌入。说到词向量，不少读者应该会立马想到Google出品的Word2Vec，大牌效应就是不一样。另外，用Keras之类的框架还有一个Embedding层，也说是将词ID映射为向量。由于先入为主的意识，大家可能就会将词向量跟Word2Vec等同起来，而反过来问“Embedding是哪种词向量？”这类问题，尤其是对于初学者来说，应该是很混淆的。事实上，哪怕对于老手，也不一定能够很好地说清楚。

这一切，还得从one hot说起...

五十步笑百步

one hot，中文可以翻译为“独热”，是最原始的用来表示字、词的方式。为了简单，本文以字为例，词也是类似的。假如词表中有“科、学、空、间、不、错”六个字，one hot就是给这六个字分别用一个0-1编码：

科	$[1,0,0,0,0,0]$
学	$[0,1,0,0,0,0]$
空	$[0,0,1,0,0,0]$
间	$[0,0,0,1,0,0]$
不	$[0,0,0,0,1,0]$
错	$[0,0,0,0,0,1]$

那么，如果表示“科学”这个词，那么就可以用矩阵

$$\begin{pmatrix}1&0&0&0&0&0\\0&1&0&0&0&0\end{pmatrix}$$

大家可能感觉到问题了，有多少个字，就得有多少维向量，假如有1万字，那么每个字向量就是1万维（常用的字可能不多，几千个左右，但是按照词的概念来看，常用的词可能就有十几万了）。于是就出来了连续向量表示，比如用100维的实数向量来表示一个字，这样就大大降低了维度，降低了过拟合的风险，等等。初学者是这样说的，不少专家也是这样说的。

然而事实是：放屁！放屁！放屁！重要的事情说三遍。

给大家出道题大家给明白了：给两个任意实数型的100阶矩阵让你算它们的乘积，可能没几个人能够算出来；可是，给你两个1000阶的矩阵，但其中一个是one hot型（每一行只有一个元素为1，其它都是0）的矩阵，让你相乘，你很快就能算出来了，不信你就试试。

看出问题来了吧？one hot矩阵是庞大，但是人家好算，你那个什么鬼实数矩阵，虽然维度小，但是算起来还麻烦呢（虽然这点计算量对于计算机来说算不了什么）！当然，更深刻的原因还在下面。

似非而是

我们真的去算一次

$$\begin{pmatrix}1&0&0&0&0&0\\0&1&0&0&0&0\end{pmatrix}\begin{pmatrix}w_{11}&w_{12}&w_{13}\\w_{21}&w_{22}&w_{23}\\w_{31}&w_{32}&w_{33}\\w_{41}&w_{42}&w_{43}\\w_{51}&w_{52}&w_{53}\\w_{61}&w_{62}&w_{63}\end{pmatrix}=\begin{pmatrix}w_{11}&w_{12}&w_{13}\\w_{21}&w_{22}&w_{23}\end{pmatrix}$$

左边的形式表明，这是一个以2x6的one hot矩阵的为输入、中间层节点数为3的全连接神经网络层，但你看右边，不就相当于在w_{ij}这个矩阵中，取出第1、2行，这不是跟所谓的字向量的查表（从表中找出对应字的向量）是一样的吗？事实上，正是如此！这就是所谓的Embedding层，Embedding层就是以one hot为输入、中间层节点为字向量维数的全连接层！而这个全连接层的参数，就是一个“字向量表”！从这个层面来看，字向量没有做任何事情！它就是one hot，别再嘲笑one hot的问题了，字向量就是one hot的全连接层的参数！

那么，字向量、词向量这些，真的没有任何创新了吗？有的，从运算上来看，基本上就是通过研究发现，one hot型的矩阵相乘，就像是相当于查表，于是它直接用查表作为操作，而不写成矩阵再运算，这大大降低了运算量。再次强调，降低了运算量不是因为词向量的出现，而是因为把one hot型的矩阵运算简化为了查表操作。这是运算层面的。思想层面的，就是它得到了这个全连接层的参数之后，直接用这个全连接层的参数作为特征，或者说，用这个全连接层的参数作为字、词的表示，从而得到了字、词向量，最后还发现了一些有趣的性质，比如向量的夹角余弦能够在某种程度上表示字、词的相似度。

对了，有人诟病，Word2Vec（CBOW）只是一个三层的模型，算不上“深度”学习，事实上，算上one hot的全连接层，就有4层了，也基本说得上小小的深度模型了。

从何而来

等等，如果把字向量当做全连接层的参数（这位读者，我得纠正，不是“当做”，它本来就是），那么这个参数你还没告诉我怎么得到呢！答案是：我也不知道怎么得来呀。神经网络的参数不是取决你的任务吗？你的任务应该问你自己呀，怎么问我来了？你说Word2Vec是无监督的？那我再来澄清一下。

严格来讲，神经网络都是有监督的，而Word2Vec之类的模型，准确来说应该是“自监督”的，它事实上训练了一个语言模型，通过语言模型来获取词向量。所谓语言模型，就是通过前n个字预测下一个字的概率，就是一个多分类器而已，我们输入one hot，然后连接一个全连接层，然后再连接若干个层，最后接一个softmax分类器，就可以得到语言模型了，然后将大批量文本输入训练就行了，最后得到第一个全连接层的参数，就是字、词向量表，当然，Word2Vec还做了大量的简化，但是那都是在语言模型本身做的简化，它的第一层还是全连接层，全连接层的参数就是字、词向量表。

这样看，问题就比较简单了，我也没必要一定要用语言模型来训练向量吧？对呀，你可以用其他任务，比如文本情感分类任务来有监督训练。因为都已经说了，就是一个全连接层而已，后面接什么，当然自己决定。当然，由于标签数据一般不会很多，因此这样容易过拟合，因此一般先用大规模语料无监督训练字、词向量，降低过拟合风险。注意，**降低过拟合风险的原因是可以使用无标签语料预训练词向量出来（无标签语料可以很大，语料足够大就不会有过拟合风险），跟词向量无关，词向量就是一层待训练参数，有什么本事降低过拟合风险？**

最后，解释一下为什么这些字词向量会有一些性质，比如向量的夹角余弦、向量的欧氏距离都能在一定程度上反应字词之间的相似性？这是因为，我们在用语言模型无监督训练时，是开了窗口的，通过前n个字预测下一个字的概率，这个n就是窗口的大小，同一个窗口内的词语，会有相似的更新，这些更新会累积，而具有相似模式的词语就会把这些相似更新累积到可观的程度。我举个例子，“忐”、“忑”这两个字，几乎是连在一起用的，更新“忐”的同时，几乎也会更新“忑”，因此它们的更新几乎都是相同的，这样“忐”、“忑”的字向量必然几乎是一样的。“相似的模式”指的是在特定的语言任务中，它们是可替换的，比如在一般的泛化语料中，“我喜欢你”中的“喜欢”，以及一般语境下的“喜欢”，替换为“讨厌”后还是一个成立的句子，因此“喜欢”与“讨厌”必然具有相似的词向量，但如果词向量是通过情感分类任务训练的，那么“喜欢”与“讨厌”就会有差异较大的词向量。

未完不待续

感觉还没说完，但好像也没有什么好说的了，希望这点文字有助于大家理解字、词向量这些概念，弄清楚one hot和Embedding的本质。有疑问或者有新的见解，欢迎留言提出。