大白话讲解word2vec构建词向量

IT生态 2018-12-20

词向量

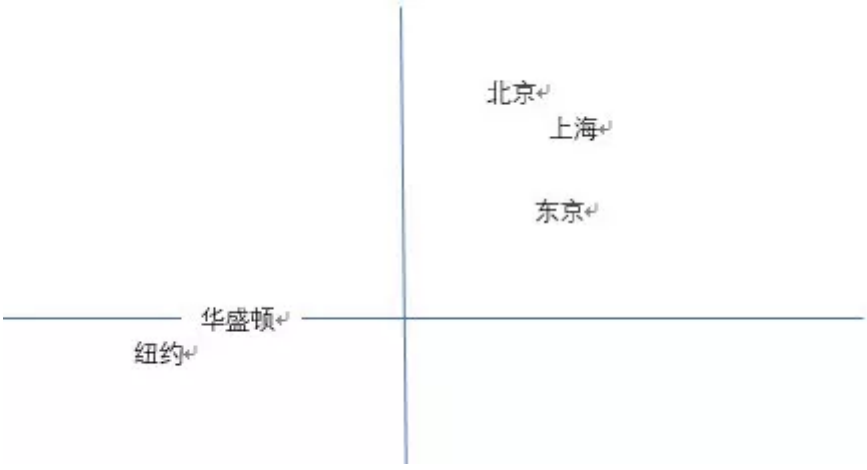
word2vec也叫word embeddings，中文名“词向量”，作用就是将自然语言中的字词转为计算机可以理解的稠密向量（Dense Vector）。在word2vec出现之前，自然语言处理经常把字词转为离散的单独的符号，也就是One-Hot Encoder。

```
杭州 [0,0,0,0,0,0,0,1,0,……, 0,0,0,0,0,0,0]
上海 [0,0,0,0,1,0,0,0,0,……, 0,0,0,0,0,0,0]
宁波 [0,0,0,1,0,0,0,0,0,……, 0,0,0,0,0,0,0]
北京 [0,0,0,0,0,0,0,0,0,……, 1,0,0,0,0,0,0]
```

比如上面的这个例子，在语料库中，杭州、上海、宁波、北京各对应一个向量，向量中只有一个值为1，其余都为0。但是使用One-Hot Encoder有以下问题。一方面，城市编码是随机的，向量之间相互独立，看不出城市之间可能存在的关联关系。其次，向量维度的大小取决于语料库中字词的多少。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。

使用Vector Representations可以有效解决这个问题。Word2Vec可以将One-Hot Encoder转化为低维度的连续值，也就是稠密向量，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。

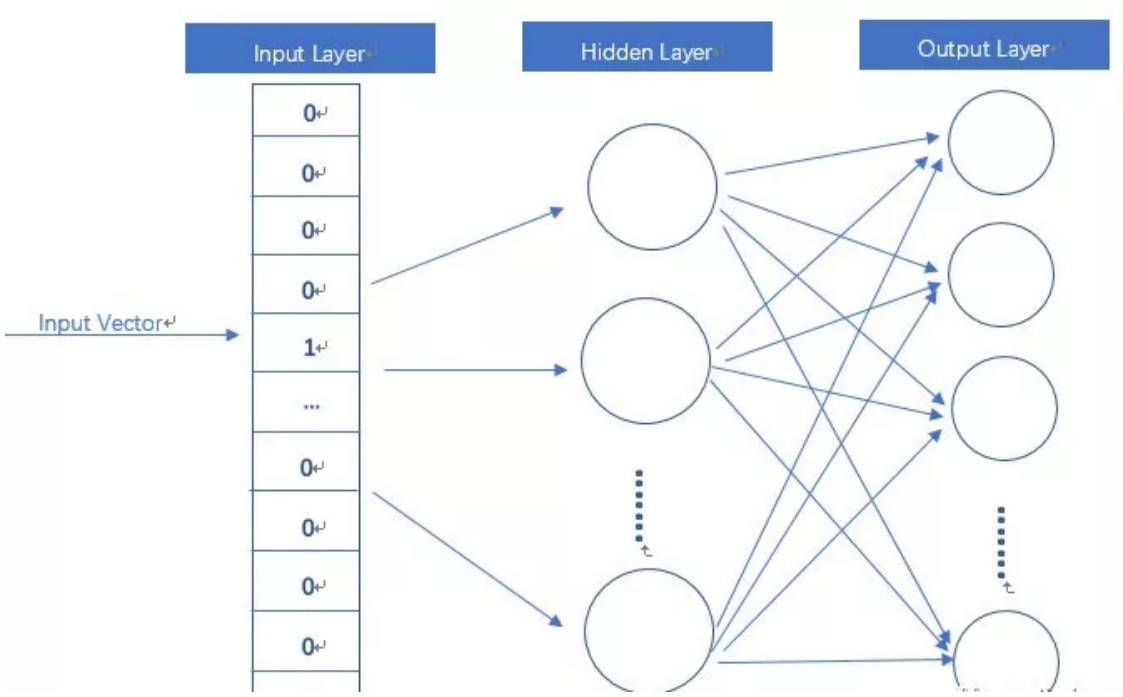
如果将embed后的城市向量通过PCA降维后可视化展示出来，那就是这个样子。



我们可以发现，华盛顿和纽约聚集在一起，北京上海聚集在一起，且北京到上海的距离与华盛顿到纽约的距离相近。也就是说模型学习到了城市的地理位置，也学习到了城市地位的关系。

模型拆解

word2vec模型其实就是简单化的神经网络。



输入是One-Hot Vector，Hidden Layer没有激活函数，也就是线性的单元。Output Layer维度跟Input Layer的维度一样，用的是Softmax回归。我们要获取的dense vector其实就是Hidden Layer的输出单元。有的地方定为Input Layer和Hidden Layer之间的权重，其实说的是一回事。

$$[0 \quad 0 \quad 0 \quad 1 \quad 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = [10 \quad 12 \quad 19]$$

CBOW与Skip-Gram模式

word2vec主要分为CBOW（Continuous Bag of Words）和Skip-Gram两种模式。CBOW是从原始语句推测目标字词；而Skip-Gram正好相反，是从目标字词推测出原始语句。CBOW对小型数据库比较合适，而Skip-Gram在大型语料中表现更好。

对同样一个句子：Hangzhou is a nice city。我们要构造一个语境与目标词汇的映射关系，其实就是input与label的关系。

这里假设滑窗尺寸为1（滑窗尺寸……这个……不懂自己google吧-_-|||）
CBOW可以制造的映射关系为：[Hangzhou,a]—>is，[is,nice]—>a，[a,city]—>nice
Skip-Gram可以制造的映射关系为(is,Hangzhou)，(is,a)，(a,is)， (a,nice)，(nice,a)，(nice,city)

训练优化

额，到这里，你可能会注意到，这个训练过程的参数规模非常巨大。假设语料库中有30000个不同的单词，hidden layer取128，word2vec两个权值矩阵维度都是[30000,128]，在使用SGD对庞大的神经网络进行学习时，将是十分缓慢的。而且，你需要大量的训练数据来调整许多权重，避免过度拟合。数以百万计的重量数十亿倍的训练样本意味着训练这个模型将是一个野兽。
一般来说，有Hierarchical Softmax、Negative Sampling等方式来解决。

简而言之，词向量技术是将词转化成为稠密向量，并且对于相似的词，其对应的词向量也相近。

一、词的表示

在自然语言处理任务中，首先需要考虑词如何在计算机中表示。通常，有两种表示方式：one-hot representation和distribution representation。

1.1离散表示（one-hot representation）

传统的基于规则或基于统计的自然语义处理方法将单词看作一个原子符号
被称作one-hot representation。one-hot representation把每个词表示为一个长向量。这个向量的维度是词表大小，向量中只有一个维度的值为1，其余维度为0，这个维度就代表了当前的词。
例如：
苹果 [0，0，0，1，0，0，0，0，0，……]
one-hot representation相当于给每个词分配一个id，这就导致这种表示方式不能展示词与词之间的关系。另外，one-hot representation将会导致特征空间非常大，但也带来一个好处，就是在高维空间中，很多应用任务线性可分。

1.2分布式表示（distribution representation）

word embedding指的是将词转化成一种分布式表示，又称词向量。分布式表示将词表示成一个定长的连续的稠密向量。

分布式表示优点:
(1)词之间存在相似关系：
是词之间存在“距离”概念，这对很多自然语言处理的任务非常有帮助。
(2)包含更多信息：

词向量能够包含更多信息，并且每一维都有特定的含义。在采用one-hot特征时，可以对特征向量进行删减，词向量则不能。

二、如何生成词向量

本小节来简单介绍词向量的生成技术。生成词向量的方法有很多，这些方法都依照一个思想：任一词的含义可以用它的周边词来表示。生成词向量的方式可分为：基于统计的方法和基于语言模型(language model)的方法。

2.1 基于统计方法

2.1.1 共现矩阵

通过统计一个事先指定大小的窗口内的word共现次数，以word周边的共现词的次数做为当前word的vector。具体来说，我们通过从大量的语料文本中构建一个共现矩阵来定义word representation。

例如，有语料如下：

I like deep learning.

I like NLP.

I enjoy flying.

则其共现矩阵如下：

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1

矩阵定义的词向量在一定程度上缓解了one-hot向量相似度为0的问题，但没有解决数据稀疏性和维度灾难的问题。

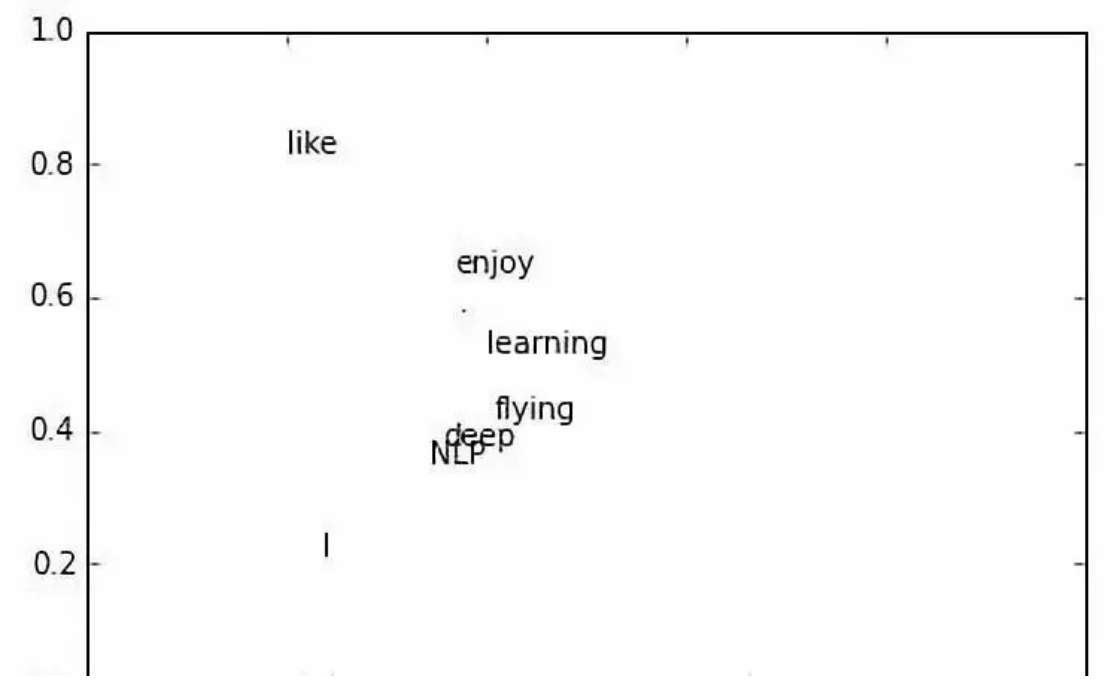
2.1.2 SVD（奇异值分解）

既然基于co-occurrence矩阵得到的离散词向量存在着高维和稀疏性的问题，一个自然而然的解决思路是对原始词向量进行降维，从而得到一个稠密的连续词向量。

对2.1.1中矩阵，进行SVD分解，得到矩阵正交矩阵U，对U进行归一化得到矩阵如下：

I	0.24	0.21	0.10	0.38	-0.18	-0.18	-0.42	-0.06
like	0.20	0.82	-0.17	0.31	0.18	-0.23	0.13	0.14
enjoy	0.37	0.64	0.16	0.00	-0.58	0.64	0.00	-0.31
deep	0.36	0.38	0.35	-0.07	0.45	0.08	0.55	-0.47
learning	0.40	0.52	-0.50	-0.43	0.35	0.16	-0.47	-0.40
NLP	0.35	0.35	-0.22	-0.19	0.13	0.49	0.21	0.66
flying	0.41	0.42	-0.40	-0.38	-0.51	-0.43	0.4	12

SVD得到了word的稠密（dense）矩阵，该矩阵具有很多良好的性质：语义相近的词在向量空间相近，甚至可以一定程度反映word间的线性关系。



2.2语言模型(language model)

语言模型生成词向量是通过训练神经网络语言模型NNLM（neural network language model），词向量做为语言模型的附带产出。NNLM背后的基本思想是对出现在上下文环境里的词进行预测，这种对上下文环境的预测本质上也是一种对共现统计特征的学习。

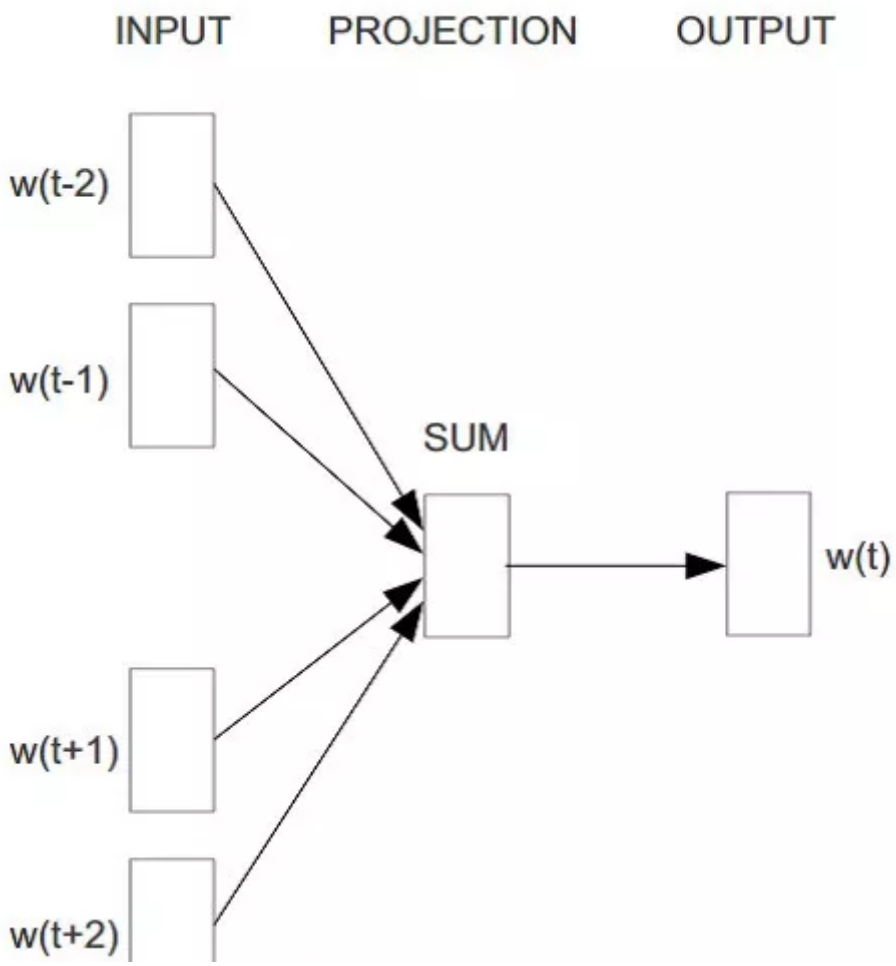
较著名的采用neural network language model生成词向量的方法有：Skip-gram、CBOW、LBL、NNLM、C&W、GloVe等。接下来，以目前使用最广泛CBOW模型为例，来介绍如何采用语言模型生成词向量。

2.2.1 CBOW（Continuous Bag-of-Word）

生成word的distribution representation的模型有很多，生成的向量效果差距不大（生成向量的效果更取决于良好的训练数据）。本节以CBOW模型中的层次化的softmax为例，介绍如何采用神经网络生成词向量。

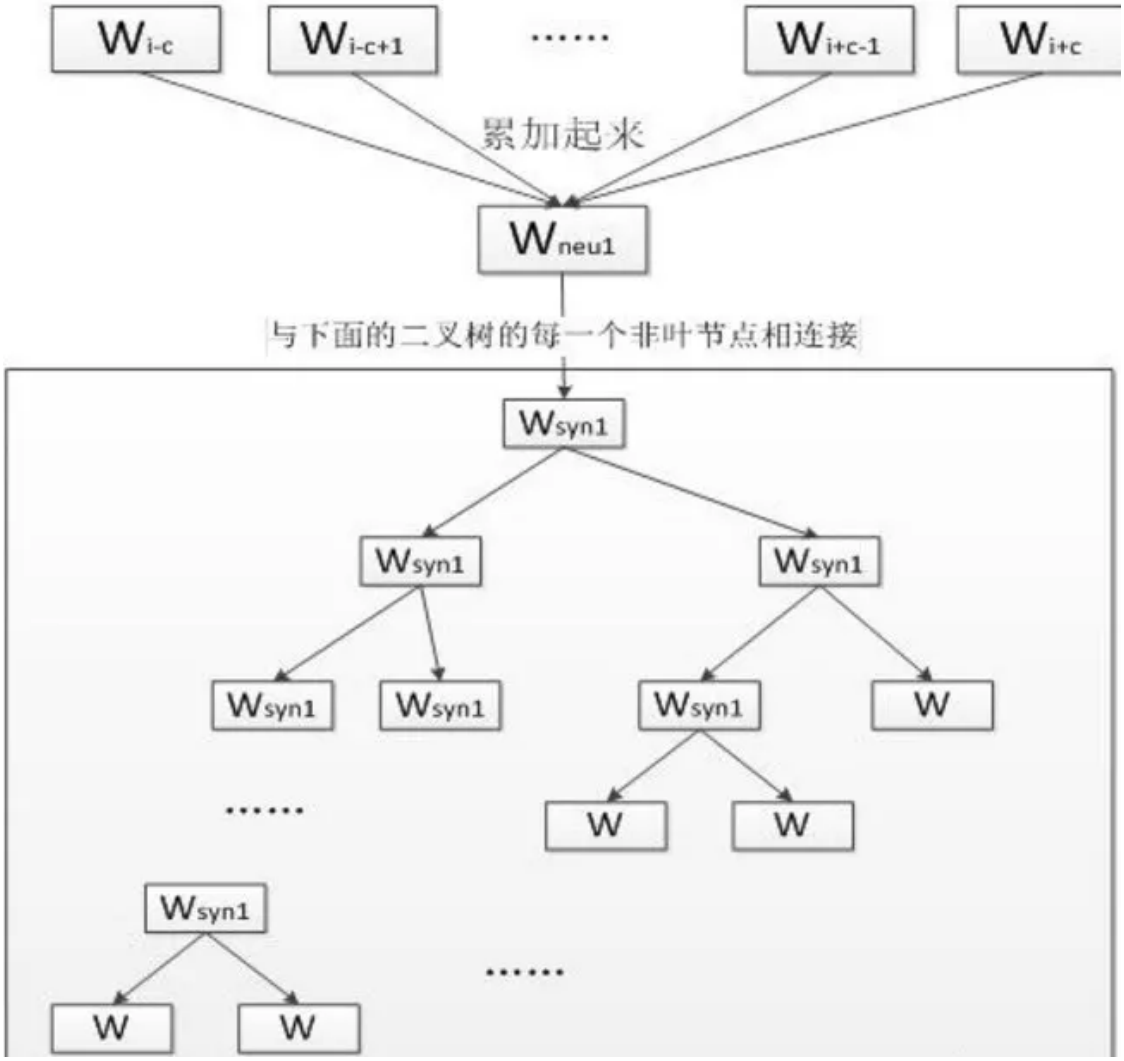
CBOW模型是预测上下文已知的情况下，当前词出现的概率。上下文的选取采用窗口方式，即只将当前词窗口范围内的词作为上下文。中心词概率公式如下：

$P(wt|wt-k,wt-(k-1)...,wt-1,wt+1,wt+2...,wt+k) = P(wt|context)$



CBOW模型去掉了最耗时的隐层，从输入层到隐层的操作是对上下文词向量进行求和(word2vec代码对以上下文词向量的加权平均来表示context)。
从隐层到输出层则利用上下文词向量和做为输入，输出的是窗口中心位置所有词出现的概率。利用softmax求中心词概率，公式表示如下：
$p(w_t|context)=e^{(-g(w_t,context))}/(\sum_{v\in V} e^{(-g(v,context))})$
其中，context表示上下文词向量和，V表示词表，wt表示中心词，g表示能量函数。由上式可看出，分母计算比较密集，时间复杂度O|V|。由上式可以看出，分母的计算复杂度与词表规模相关，当语料较大时，计算变的非常耗时。解决这一问题是对词进行多分类/聚类，例如，5000词的语料，聚成100类，则时间复杂度由5000缩减至50。

CBOW采用了一种Hierarchical softmax技术，利用哈夫曼树对词表进行分类，用一连串的二分类来近似多分类。



加入哈夫曼树后，中心词的概率可表示入下：

$$p(w_t \mid context)=\prod_{i=1}^{k} [\![p(d_i \mid q_i,context)]\!]$$
$$=\prod_{i=1}^{k} [\![\sigma(v_{context}^T * q_i)]\!]$$

其中，$v_{context}^T$为窗口内词的词向量的加权平均，q_i为结点i的结点向量，σ(*)为神经网络激活函数。从公式中可以看出，中心词的概率是从根节点到词所在的叶节点所走过路径概率的乘积。

对中心词概率做极大似然估计，可得到模型的损失函数。损失函数如下：

$$Loss=-Likelihood=-(1-code[j])\log \sigma(v_{context}^T * q_i)-code[j]\log(1-\sigma(v_{context}^T * q_i))$$

code[j]为结点j的哈夫曼编码。有了损失函数，就可以通过梯度下降法求得词向量。注意，损失函数中有2个带求参数，分别为〖词向量（v〗_context^T）和结点向量〖（q〗_i）。

可以看出，CBOW模型是一个语言模型。在训练好后，语言模型的参数$v_{context}^T$作为副产出，得到词向量。

三．词向量的训练

本节将分享我使用词向量的一些经验。

3.1词向量效果的影响因素

（1）增加词向量的维度能够增加词向量效果。

（2）在同一领域语料下，语料越多越好，增加不相关领域语料将会降低词向量效果。

（3）大的上下文窗口学到的词向量更反映主题信息，而小的上下文窗口学到的词向量更反映词的功能和上下文语义信息。

（4）语料的纯度越高（杂质少），词向量效果越好。因此，在利用语料训练词向量时，进行预处理能够提高词向量的效果。

3.2词向量的歧义问题

例如，词”苹果”可以指代水果也可以指代苹果手机。在对“苹果”
进行训练时，将会对其对应的词向量向两个方向拉伸，进而造成词向量歧义。
词向量的歧义将会对词向量的应用效果产生影响。例如，对苹果进行序列标注，指代手机的苹果应打上品牌词标签，而指代水果的苹果对应打上产品词标签。对同一个词打上多种不同的标签而采用同一个词向量，将降低词性标注的效果。
通常解决这一问题是对词向量进行聚类，以多个向量来表示同一个词。
例如，在RNNs分词项目时，发现字向量的歧义问题将影响分词效果。因此，我按品类对字进行了聚类，得到的聚类效果如下：

- 歧义字向量效果:

红色的红：

红

Word	Cosine distance
瑰	0.594500
色	0.541844
店	0.503747
女	0.497063
黄	0.476341
杏	0.436897
码	0.430535
橙	0.427505
敬	0.425439
紫	0.420456
橘	0.417114
款	0.416879
吧	0.415589
黑	0.415420
长	0.401261

红米的红：

红@#@9987

y: 783

Word	Cosine distance
小@#@9987	0.837753
班@#@9987	0.643777
糯@#@9987	0.631457
纳@#@9987	0.595923
妮@#@9987	0.505155
5@#@9987	0.490820
2@#@9987	0.474556
都@#@9987	0.459761
瑰@#@9987	0.458887
捞@#@9987	0.456077
森@#@9987	0.438582
3@#@9987	0.435415
壳@#@9987	0.418292
手@#@9987	0.417590
套@#@9987	

左边是颜色中红色的红，可以看出它与其它颜色的距离较近。右边是手机品类红米手机的“红”，可以看到它最相似的词是“小”字，因为在京东的商品title里，手机品类中红米手机和小米手机出现的次数较多，进而使得“红”字和“小”字的语境相似。

3.3 词向量其它

还可以利用NNLM方法，将我们感兴趣的其它实体生成向量。例如，我曾利用word2vec将每个sku(商品id)embedding成向量。我们将每个用户某个月购买的母婴类商品按序进行排列做为一条训练样本，例如：

其中，第一列user_id +’#’+用户婴儿年龄。其余列为用户购买的sku集合，并且购买的sku按购买时间顺序排列。
我们将训练样本输入到word2vec中进行训练，得到每个sku的向量表示。通过这种训练样本的构建方式，使用户在相同年龄段（婴儿）购买的商品相似。例如，在我们的向量中，如下两个商品相似：

我们利用sku向量做为特征，训练模型预测用户年龄，准确率达90%。

四．总结
深度学习模型具有比传统模型更强的特征抽取能力。在自然语言处理中，传统统计特征包含的信息量过少，这也一直限制着深度学习在自然语言处理中的应用。词向量由于包含了更丰富的信息，使得深度学习能够处理绝大多数自然语言处理应用。
词向量虽然为深度学习在自然语言处理领域带来了希望，但目前词向量仍有许多不完善的地方，例如：
虽然知道词向量比统计特征包含更多信息，但并不知道词向量包含了哪些信息（如每维特征代表的意义）。
词向量的训练采用无监督方式，不能很好的利用先验信息。
词向量是神经网络语言模型的副产物，其损失函数不是由具体应用构建。
因此，不是词向量训练的越好，应用效果就越好。

随着词向量研究的深入，这些问题都将会得到解决。深度学习在自然语言处理领域的前景将会更加光明。