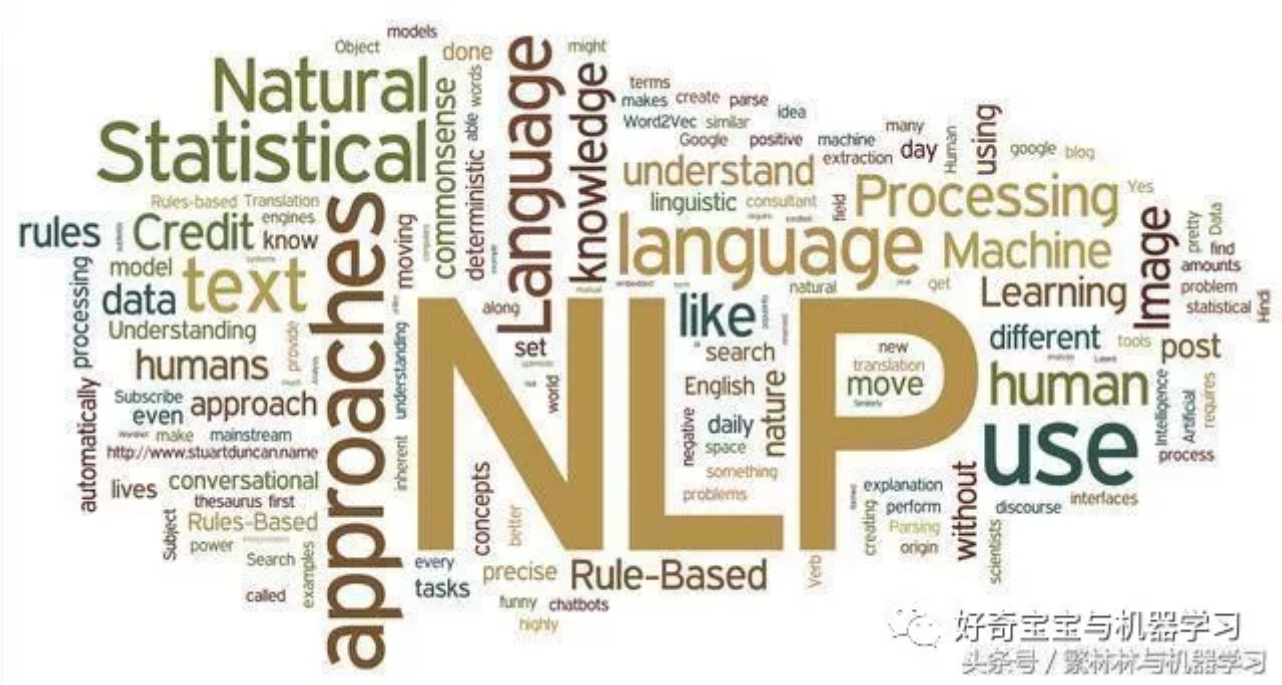
# 自然语言处理-第五期-Word2Vec

原创 繁林林 繁林林与机器学习 2018-07-12

## 背景




上期我们介绍了Bag of Word，它可以将word转换为one-hot vector，即【0，0，0，1】只有一个纬度为1，其他为0 的vector。

也提到了其最大的一个缺点，即可能造成Sparsity（过度离散）的问题。对于该问题Word2Vec给出了一个有意思的解法。

相信大家肯定有类似的经验，就是搜索的时候，会出现一些同义词也被搜索出来，比如movie和film。 这其实是一件很神奇的事情，因为人类语言是symbol，到底是如何做到的呢？ 希望大家在读完这篇文章之后，能够得到一些答案。

除此之外，本篇还会涉及到神经网络的运用，如果有不清楚的同学可以看第四期或者我的另一片文章“机器学习的hello world”，如果还有问题，可以留言，我会尽量答复。

## Word2Vec

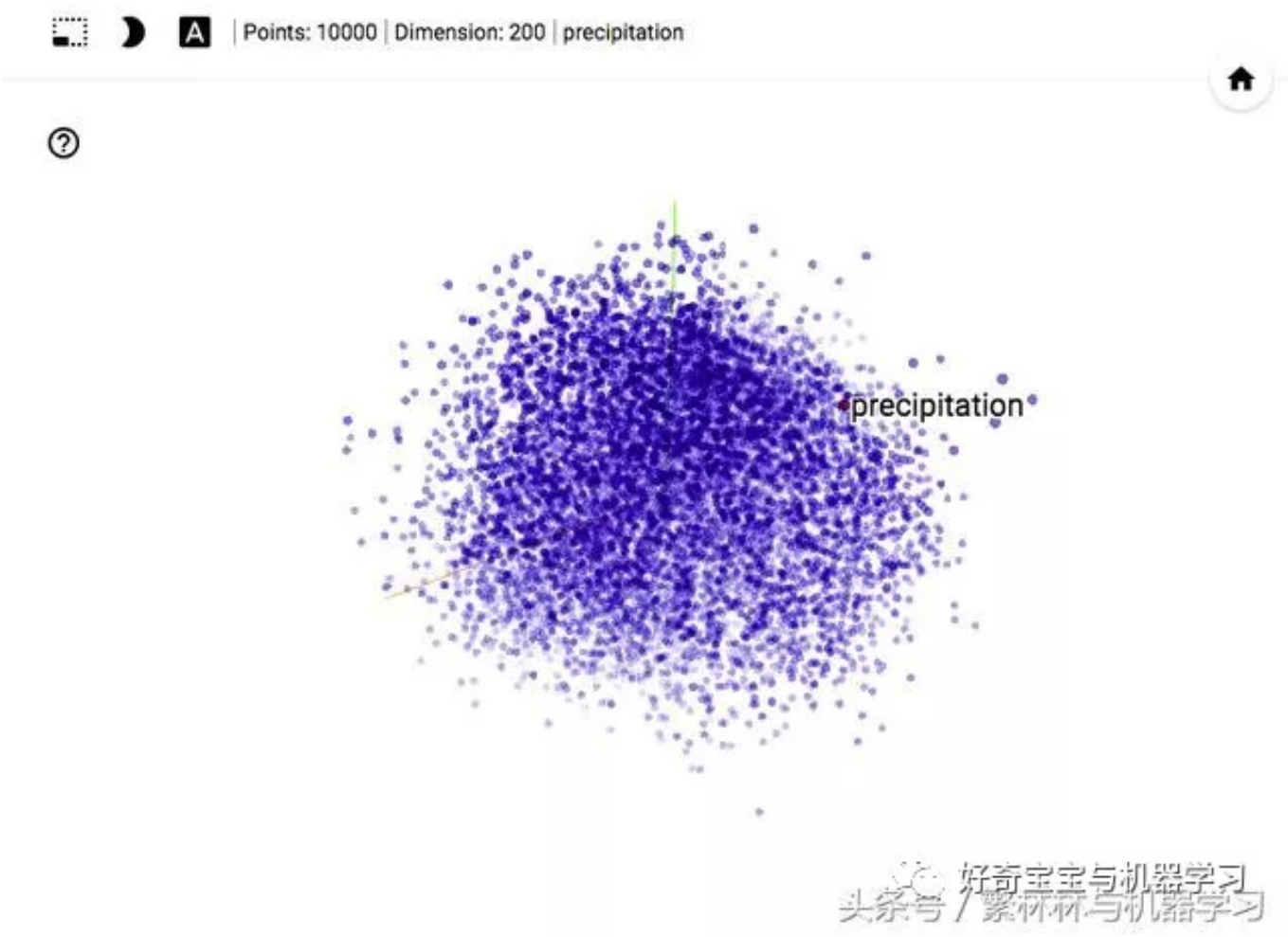
### 面对的问题及目标

我们的目标是将word向量化。bag of word也是向量化，只是之前的问题很明显。

其实我们只要不转化为one-hot vector。比如只要100个纬度的向量，每个纬度可以为小数自然数。自然其能承载的数量会大幅增加。

【0.1，0.9，8.9，11，7.5】最现实的问题是，**我们如何决定每个向量的数值，我们处理的规则是什么呢?**

在讲Skip Gram之前插个图，下面这个图是Google AI experiment 的高纬度可视化的网站，大家可以搜索AI experiment google tensorflow就能找到。在网站上做很多有趣的调整。




## Word2Vec内在逻辑

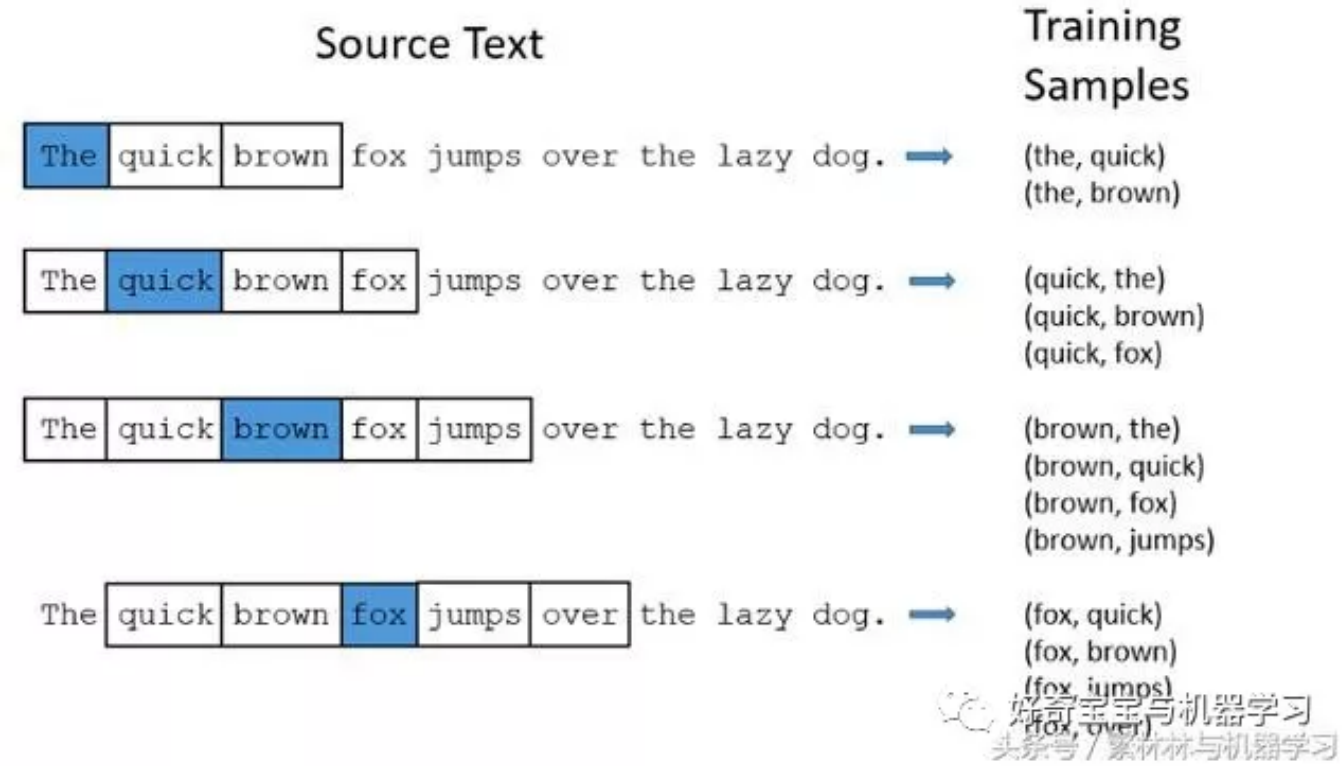
Word2Vec有好几种模型，我们今天关注其最普遍的Skip Gram模型。

我们再重复一遍问题，我们的目标是讲word的Vector纬度降低，但是How？这个问题很重要也很难，大家如果凭空想下，实际上你并没有什么原则来处理这些word。

Skip Gram提供来一种思路，就是 取一个中心词，然后预测其前后几个位子的词。如下面这个图，'the' 前后两位的词分别是 quick 和brown。

Skip Gram的思路就是训练一个神经网络来预测 中心词的附近词。

当然我们的训练需要训练样本，Skip Gram的训练样本就是从普通文本上切下来的一段一段文本（中心词不断右移）





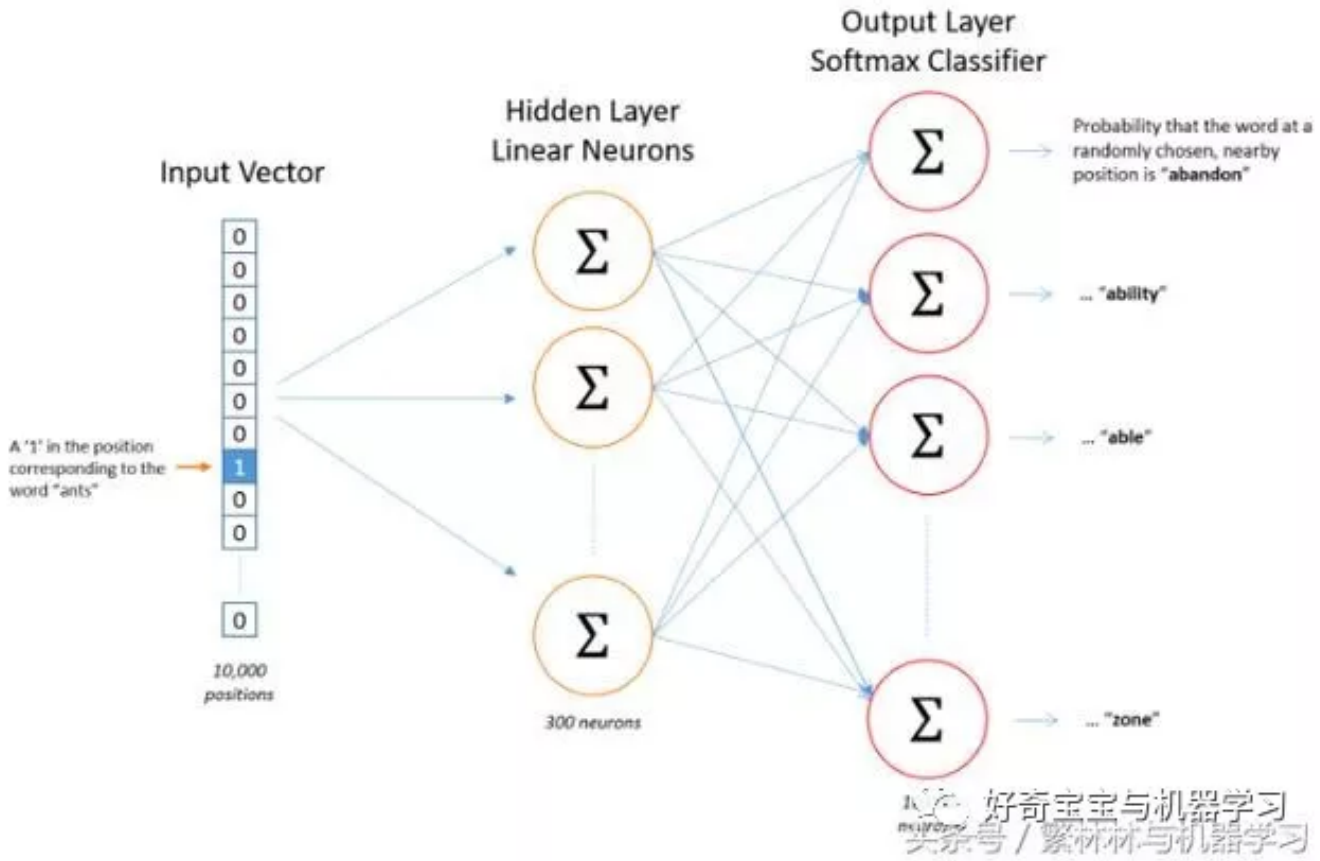

**神经网络结构**

Skip Gram的神经网络结构大概如下面所示，我们讲分输入层，隐藏层，输出层解释。

输入层，首先Skip Gram还是在处理word vector，所以肯定绕不开 one-hot vector，第一步也是建立一个词库，比如有10000个不同的单词，那么就是一个10000纬度的向量；

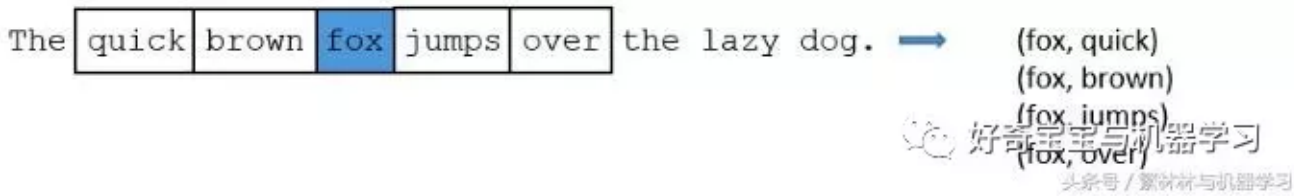
隐藏层，隐藏层的神经元个数可以自己设置，该层的个数就是下降后的向量纬度；

输出层，Skip Gram的输出层结构和输入层一样，也是10000个神经元。不同的是激励函数不同，其作用就是将每个神经元的输出变成概率（出现该词的概率）




以下面这个图片为例子，假设只有句子中的9个单词。 那么Fox中心词的向量是【0，0，0，1，0，0，0，0，0】

训练样本是（X=【0，0，0，1，0，0，0，0，0】，y=【0，1，1，0，1，1，0，0，0】）




经过神经网络训练，神经网络输出的结果，逐步与训练样本的实际值相近。

神奇的地方来了，假设我们隐藏层设置的是100个神经元。在训练完成之后，当我们输入一个10000纬度向量时，隐藏层会输出一个100纬度的向量。。。 这就完成了纬度下降。。。。

**总结下**

有点绕，总结下。

首先，Skip Gram基于中心词设计了一套训练样本（X，y），其逻辑就是给你中心词，你要能预测出周围的词；

然后，通过神经网络设计如：输入层10000，隐藏层300，输出层10000；

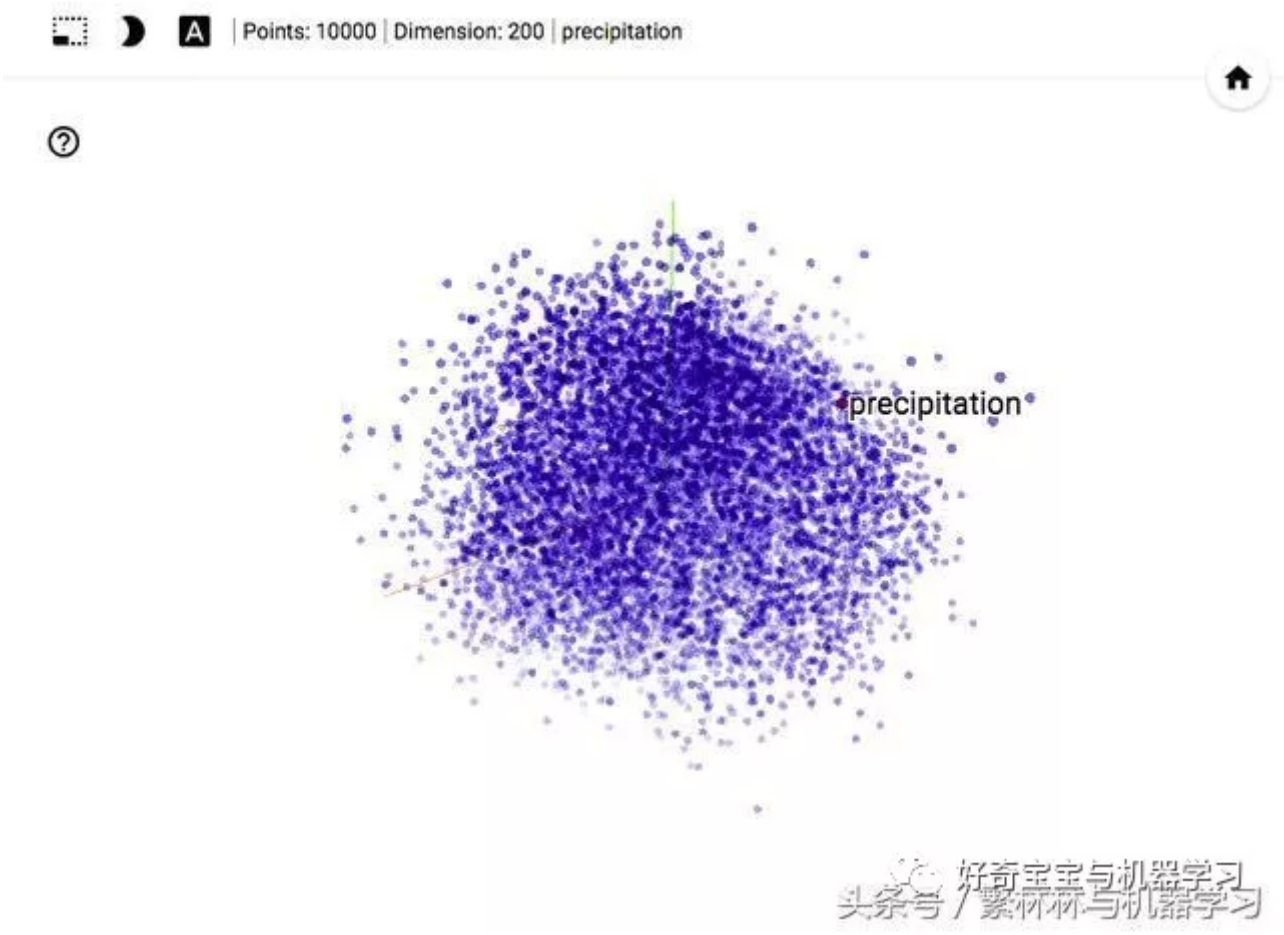
最后，在神经网络训练完成之后，我再将【1，0，0，0.....】输入进去，隐藏层会输出一个300纬度的向量。

铛铛，这就完成了word的将为，成功的将word转化为300纬度的向量。

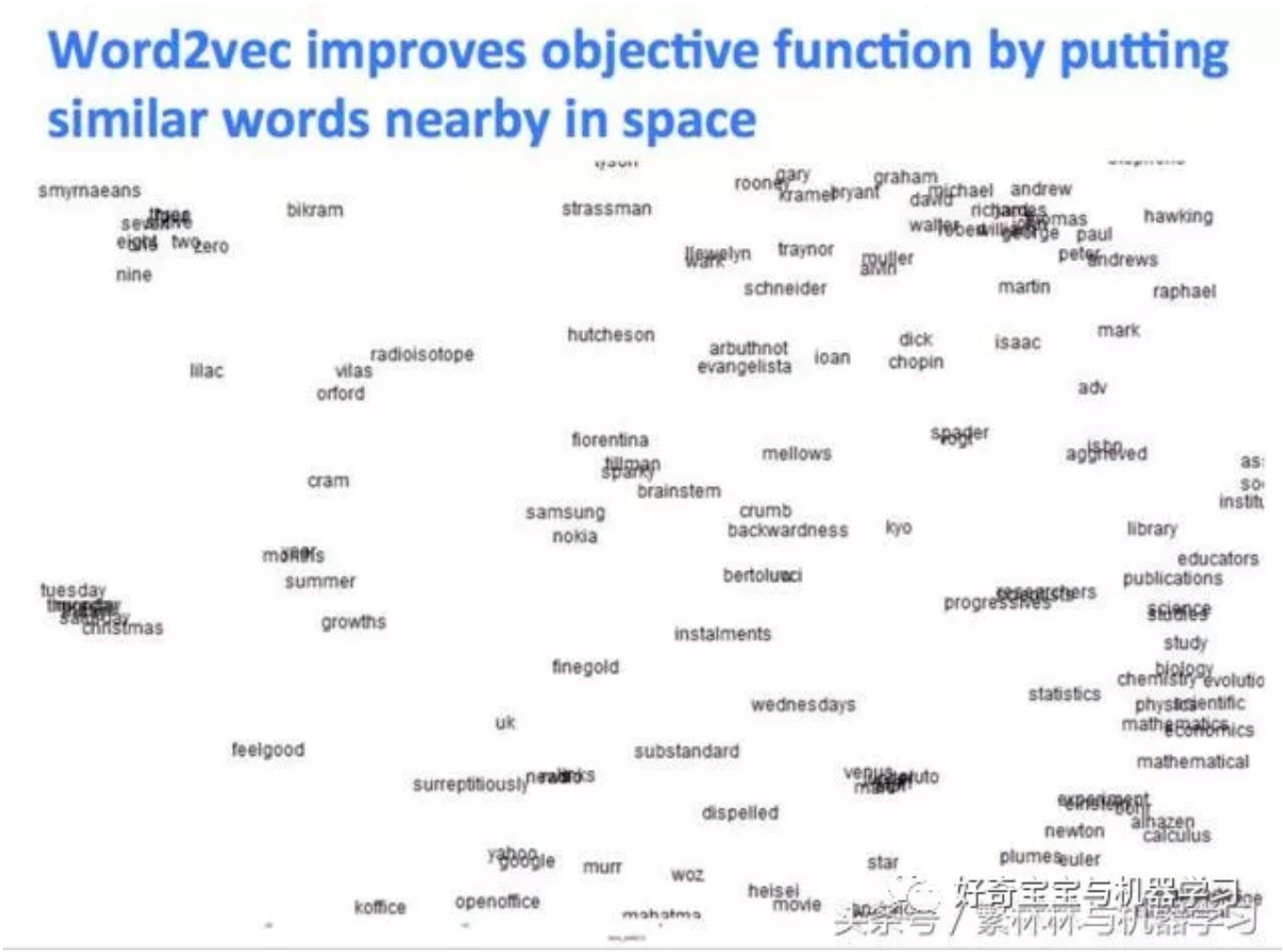
## 结果

实际上训练的时候，样本量会非常大。比如从维基百科取很多文本来训练。我的电脑无法搞定这些操作，所以Skip Gram的结果我将引用Stanford公开课CS224 的结果。

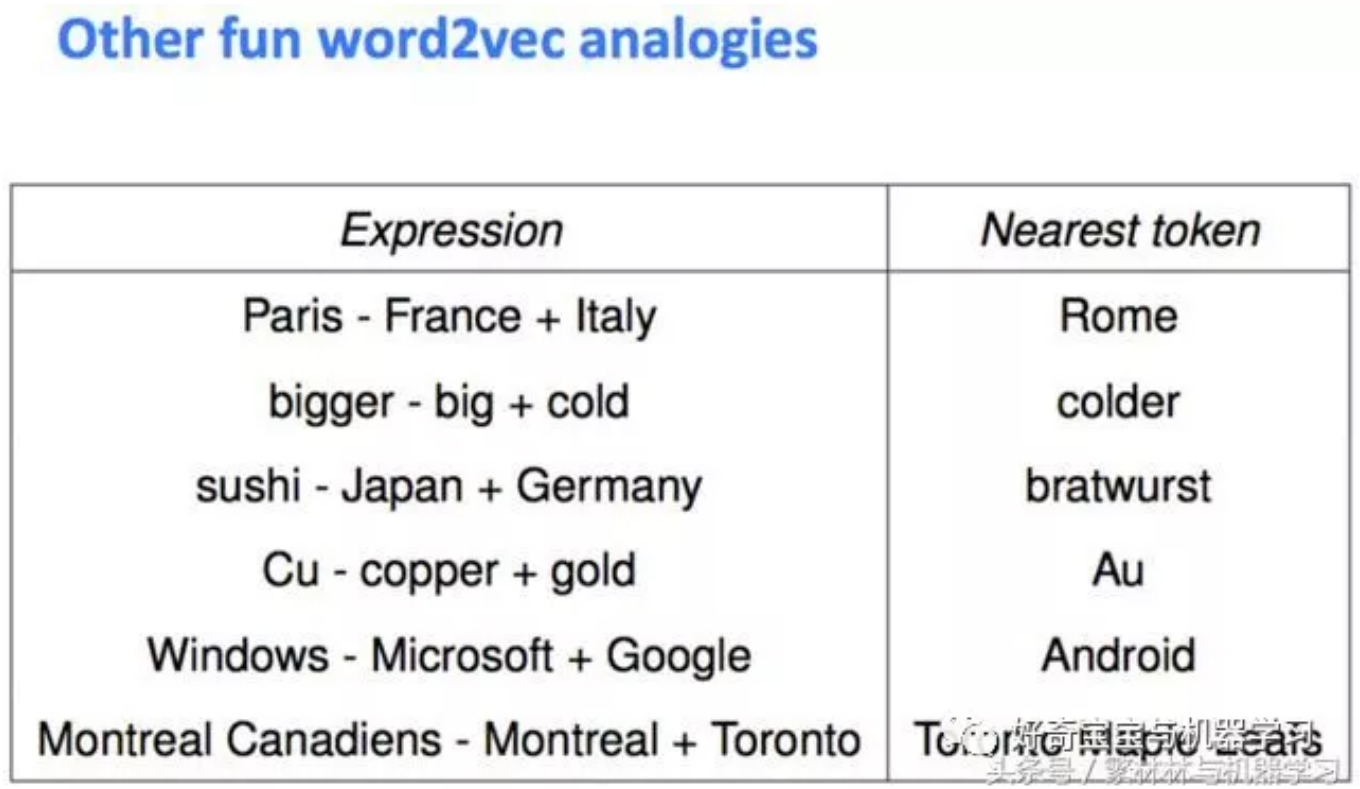
首先降纬后，Skip Gram300纬度的向量可以用PAC 3D表现出来，得到一张前面类似的图。




除此之外，更有意思的是，很多同义词会聚合在一起，如下面的图将one two three等数字都聚合在一起了。




另外很有意思的是，也能得到有意思的向量差，比如： Paris-France+Italy 最接近 Rome



其实，这种近义词在算法设计时并没有考虑。也没办法用数学证明。 但是，也可以推断某些词 是很频繁的出现在一起的。 比如Happy birthday 之类的，这些词义上的趋势实际在 维基百科的文本中就有隐藏。

## 下期预告

总之呢，这个算法还是蛮有意思的，也很直接。最终的结果也很有意思。 最终他也实现了Word的降纬。

这期已经是NLP的第五期了，一直在讲的是单词的处理。 最终我们实现了，可以使用的word vector。 那么下一期，我们将介绍第一个真正处理这些单词向量的算法。

原创码字不易，感兴趣的朋友请关注～