# word2vec 入门基础（一）

[**word2vec 入门基础（一）**](https://www.cnblogs.com/tina-smile/p/5176441.html)

# <http://www.cnblogs.com/tina-smile/p/5176441.html>

## 一、基本概念

word2vec是Google在2013年开源的一个工具，核心思想是将词表征映  
射为对应的实数向量。

目前采用的模型有一下两种

* CBOW(Continuous Bag-Of-Words，即连续的词袋模型)
* Skip-Gram

项目链接：<https://code.google.com/archive/p/word2vec>

## 二、背景知识

### 词向量

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。这样做的初衷就是机器只认识0 1 符号，换句话说，在自然语言处理中，要想让机器识别语言，就需要将自然语言抽象表示成可被机器理解的方式。所以，词向量是自然语言到机器语言的转换。

词向量有一下两种

* One-hot Representation

向量的长度为词典的大小，向量的分量只有一个 1，其他全为 0， 1 的位置对应该词在词典中的位置，例如

“话筒”表示为 [0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 ...]

“麦克”表示为 [0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 ...]

优点： 如果使用稀疏方式存储，非常简洁，实现时就可以用0,1,2,3,...来表示词语进行计算，这样“话筒”就为3，“麦克”为8.

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.任何两个词都是孤立的，存在语义鸿沟词（任意两个词之间都是孤立的，不能体现词和词之间的关系）。

* Distributional Representation

最早是 Hinton 于 1986 年提出的，可以克服 one-hot representation的缺点。解决“词汇鸿沟”问题，可以通过计算向量之间的距离（欧式距离、余弦距离等）来体现词与词的相似性  
。

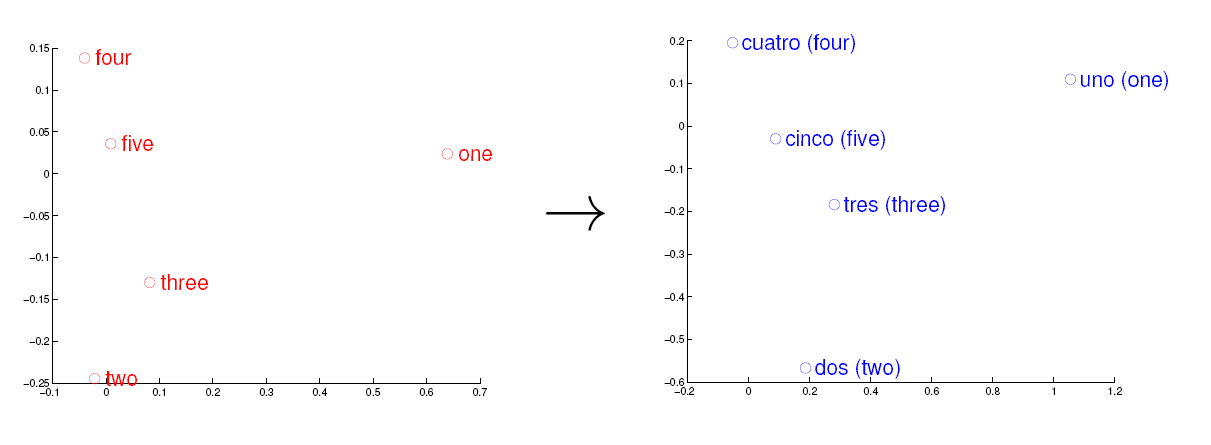
其基本想法是直接用一个普通的向量表示一个词，这种向量一般长成这个样子：[0.792, −0.177, −0.107, 0.109, −0.542, ...]，常见维度50或100。

优点：解决“词汇鸿沟”问题

缺点：训练有难度。没有直接的模型可训练得到。所以采用通过训练语言模型的同时，得到词向量 。

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。值得注意的是，**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%。

论文<http://arxiv.org/pdf/1309.4168.pdf>在介绍算法工作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，这说明两种不同语言对应向量空间的结构之间具有相似性，从而进一步说明了在词向量空间中利用距离刻画词之间相似性的合理性。

### 语言模型

语言模型其实就是看一句话是不是正常人说出来的。意义在于机器翻译、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。

用数学符号描述为：给定一个字符串"w1,w2,...,wt",计算它是自然语言的概率P(w1,w2,…,wt)。w1到wT 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,⋯*wT*)=p(*w*1)p(*w*2|*w*1)p(*w*3|*w*1,*w*2)⋯p(*wt*|*w*1,*w*2,⋯*wT*−1)

上面概率公式的意义为：第一次词确定后，看后面的词在前面次出现的情况下出现的概率。

例如，有个句子“大家喜欢吃苹果”，一共四个词"大家,喜欢,吃,苹果"

P(大家，喜欢，吃，苹果)=p(大家)p(喜欢|大家)p(吃|大家,喜欢)p(苹果|大家,喜欢,吃)

p(大家)表示“大家”这个词在语料库里面出现的概率；

p(喜欢|大家)表示“喜欢”这个词出现在“大家”后面的概率;

p(吃|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|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，因为在语料库里面很少出现。如果出现的概率高，就说明是一句自然语言

为了表示简单，上面的公式用下面的方式表示

p(s)=p(*w*1,*w*2,⋯*wT*)=∏*i*=1*Tp*(*wi*|*Contexti*)

其中，如果Contexti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。

现有模型有

* N-gram模型
* N-pos模型
* ...
* Bengio的NNLM
* C&W 的 SENNA
* M&H 的 HLBL
* Mikolov 的 RNNLM
* Huang 的语义强化

### N-gram 模型

接下来说 *p*(*wi*|*Contexti*)

的计算方法，上面看的是跟据这句话前面的所有词来计算，这样计算就很复杂，像上面那个例子得扫描四次语料库，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。这样计算开销太大。  
可以想到的优化方法就是提前将*p*(*wi*|*Contexti*) 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 ∑*i*=11000*i*!

个情况需要统计，对于计算机来说，这根本不可能。

所以诞生了N-gram模型是大词汇连续语音识别中常用的一种语言模型，对中文而言，我们称之为汉语语言模型（CLM, Chinese Language Model）。汉语语言模型利用上下文中相邻词间的搭配信息，在需要把连续无空格的拼音、笔画，或代表字母或笔画的数字，转换成汉字串（即句子）时，可以计算出最大概率的句子，从而实现从到汉字的自动转换，无需用户手动选择，避开了许多汉字对应一个相同的拼音（或笔画串、数字串）的重码问题。

我们给定一个词，然后猜测下一个词是什么。当我说“艳照门”这个词时，你想到的下一个词时什么？我想大家很有可能会想到“陈冠希”，基本上不会有人想到“陈志杰”吧。N-Gram模型的主要思想就是这样的。

上面的context都是这句话中这个词前面的所有词作为条件的概率，N-gram就是只管这个词前面的n-1个词，加上它自己，总共n个词，计算*p*(*wi*|*Contexti*)

只考虑用这n个词来算，换成数学的公式来表示，就是 p(*wi*|*Contexti*)=p(*wi*|*wi*−*n*+1,*wi*−*n*+2,⋯,*wi*−1)

一般来说，n取2或者3

假设词表中词的个数 |V| = 20,000 词，那么有下面的一些数据。



照图中的数据看去，取n=3是目前计算能力的上限了。在实践中用的最多的就是bigram和trigram了，而且效果也基本够了。

如果一个词的出现仅依赖于它前面出现的一个词，那么我们就称之为bigram。即

P(T) = P(W1W2W3…Wn)=P(W1)P(W2|W1)P(W3|W1W2)…P(Wn|W1W2…Wn-1)≈P(W1)P(W2|W1)P(W3|W2)…P(Wn|Wn-1)

如果一个词的出现仅依赖于它前面出现的两个词，那么我们就称之为trigram。

那么我们怎么得到P(Wn|W1W2…Wn-1)呢？一种简单的估计方法就是最大似然估计(Maximum Likelihood Estimate）了。即P(Wn|W1W2…Wn-1) = (C(W1 W2…Wn))/(C(W1 W2…Wn-1))  
下面我们用bigram举个例子。假设语料库总词数为13,748  
  


 P(I want to eat Chinese food)

=P(I)P(want|I)P(to|want)P(eat|to)P(Chinese|eat)\*P(food|Chinese)

=0.251087/3437786/1215860/325619/938\*120/213

=0.000154171

N-gram模型也会有写问题，总结如下：

1、n不能取太大，取大了语料库经常不足，所以基本是用降级的方法

2、无法建模出词之间的相似度，就是有两个词经常出现在同一个context后面，但是模型是没法体现这个相似性的。

3、有些n元组（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，解决的方法主要是两种：平滑法（基本上是分子分母都加一个数）和回退法（利用n-1的元组的概率去代替n元组的概率）

该博文主要是前段时间学习word2vec的一个总结，大部分都是网上资料，主要是做了个整理。算是做个备份，方便以后复习之类的。在此感谢前辈们的资料~~~

该部分仅仅回顾一些基本概念和背景知识，明天继续更。

# [word2vec 入门（二）使用教程篇](https://www.cnblogs.com/tina-smile/p/5178549.html)

# <http://www.cnblogs.com/tina-smile/p/5178549.html>

### word2vec

要解决问题： 在神经网络中学习将word映射成连续（高维）向量，这样通过训练，就可以把对文本内容的处理简化为K维向量空间中向量运算，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。

一般来说， 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。另外还有其向量的加法组合算法。官网上的例子是 ：

vector('Paris') - vector('France') +  
vector('Italy') ≈vector('Rome')， vector('king') - vector('man') + vector('woman') ≈  
vector('queen')

但其实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。

### 快速入门

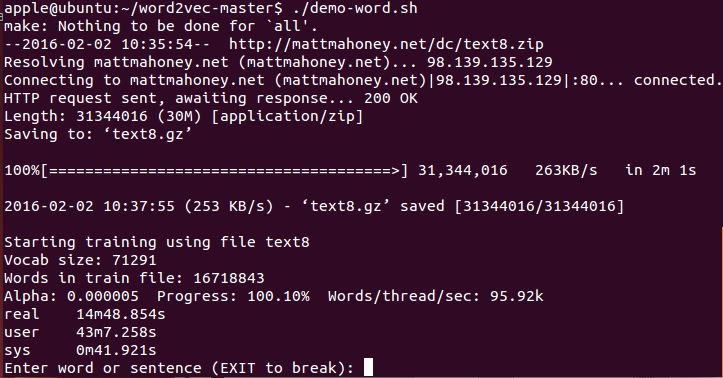
1、从http://word2vec.googlecode.com/svn/trunk/ 下载所有相关代码：  
一种方式是使用svn Checkout，可加代理进行check。  
另一种就是export to github，然后再github上下载，我选择第二种方式下载。

2、运行make编译word2vec工具：（如果其中makefile文件后有.txt后缀，将其去掉）在当前目录下执行make进行编译，生成可执行文件(编译过程中报出很出Warning，暂且不管)；

3、运行示例脚本：./demo-word.sh 看一下./demo-word.sh的内容，大致执行了3步操作

* 从http://mattmahoney.net/dc/text8.zip 下载了一个文件text8 ( 一个解压后不到100M的txt文件，可自己下载并解压放到同级目录下)；
* 使用文件text8进行训练，训练过程比较长；
* 执行word2vec生成词向量到 vectors.bin文件中，（速度比较快，几分钟的事情）

在demo-word.sh中有如下命令

运行结果如图：  


time ./word2vec -train text8 -output vectors.bin -cbow 1 -size 200 -window 8 -negative 0 -hs 1 -sample 1e-4 -threads 20 -binary 1 -iter 15

以上命令

-train text8 表示的是输入文件是text8

-output vectors.bin 输出文件是vectors.bin

-cbow 1 表示使用cbow模型，默认为Skip-Gram模型

-size 200 每个单词的向量维度是200

-window 8 训练的窗口大小为5就是考虑一个词前八个和后八个词语（实际代码中还有一个随机选窗口的过程，窗口大小小于等于8）

-negative 0 -hs 1不使用NEG方法，使用HS方法。-

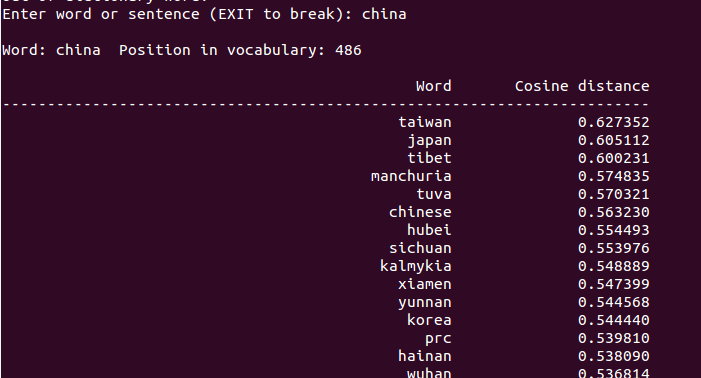
sampe指的是采样的阈值，如果一个词语在训练样本中出现的频率越大，那么就越会被采样。

-binary为1指的是结果二进制存储，为0是普通存储（普通存储的时候是可以打开看到词语和对应的向量的）

-iter 15 迭代次数

demo-word.sh中最后一行命令是./distance vectors.bin

该命令是计算距离的命令，可计算与每个词最接近的词了：

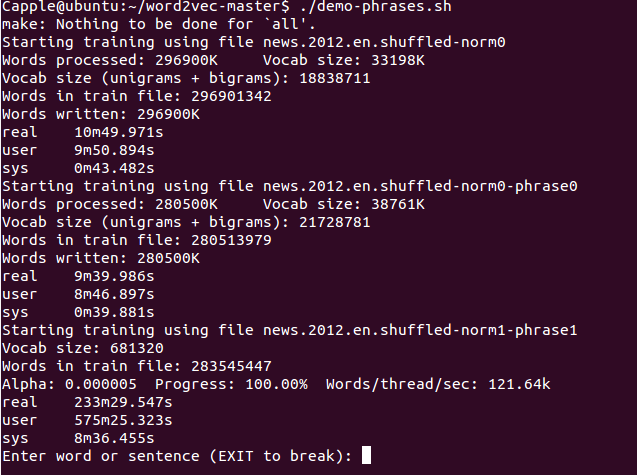


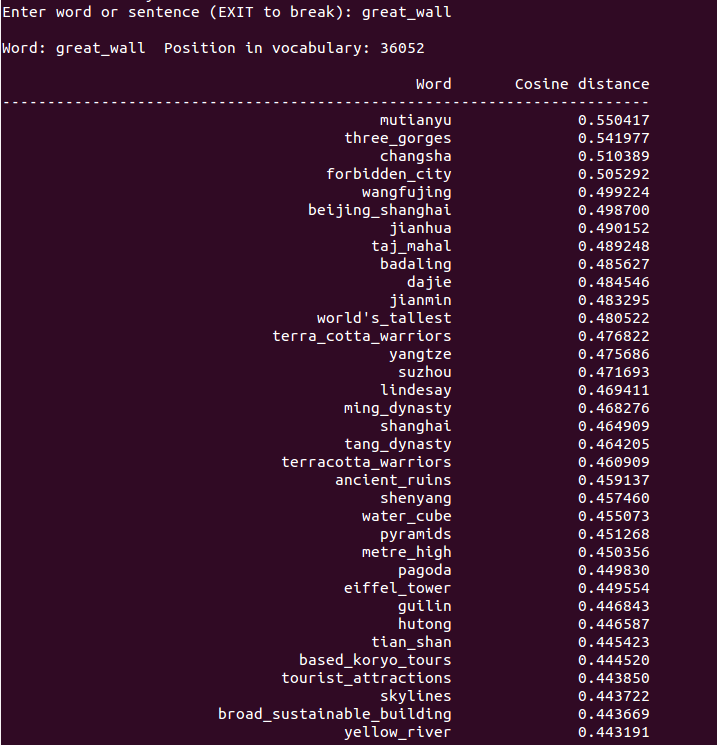
word2vec还有几个参数对我们比较有用比如-alpha设置学习速率，默认的为0.025. –min-count设置最低频率，默认是5，如果一个词语在文档中出现的次数小于5，那么就会丢弃。-classes设置聚类个数，看了一下源码用的是k-means聚类的方法。要注意-threads 20 线程数也会对结果产生影响。

* 架构：skip-gram（慢、对罕见字有利）vs CBOW（快）
* 训练算法：分层softmax（对罕见字有利）vs 负采样（对常见词和低纬向量有利）
* 欠采样频繁词：可以提高结果的准确性和速度（适用范围1e-3到1e-5）
* 文本（window）大小：skip-gram通常在10附近，CBOW通常在5附近

4、运行命令 ./demo-phrases.sh：查看该脚本内容，主要执行以下步骤：

* 从<http://www.statmt.org/wmt14/training-monolingual-news-crawl/news.2012.en.shuffled.gz> 下载了一个文件news.2012.en.shuffled.gz ( 一个解压到1.7G的txt文件，可自己下载并解压放到同级目录下)；
* 将文件中的内容拆分成 phrases，然后执行./word2vec生成短语向量到 vectors-phrase.bin文件中（数据量大，速度慢，将近半个小时），如下：

最后一行命令./distance vectors-phrase.bin，一个计算word相似度的demo中去，结果如下：  




结果好坏跟训练词库有关。

番外：  
如果需要中文语料库，推荐使用维基的或者搜狗（[http://www.sogou.com/labs/dl/ca.html](http://www.cnblogs.com/tina-smile/p/5178549.html)），中文分词可使用结巴分词，我觉得很好用。然后进行训练，因为英文不用分词，所以上述过程不涉及分词。

本文主要偏应用，讲解一个例子，便于对word2vec有一个初步了解，后续再更原理。