1. 为什么使用word2vec？还有没有其他的方法？

2. word2vec中的参数有哪些？

3. 介绍一下word2vec把？

4. 单词拼写错误检查？(优化)

5. 为什么使用了ElasticSearch之后还要用其他的东西？

6. 写爬虫中遇到了哪些困难？

7. 使用word2vec量化了为什么还要使用TFIDF？

8. 有没有对ES优化？ BM/TFIDF

9. 专有名词检查？如何产生专有名词？（优化）

10. 同义词检查？（优化）

11. 意图识别？问候！称赞！问题！

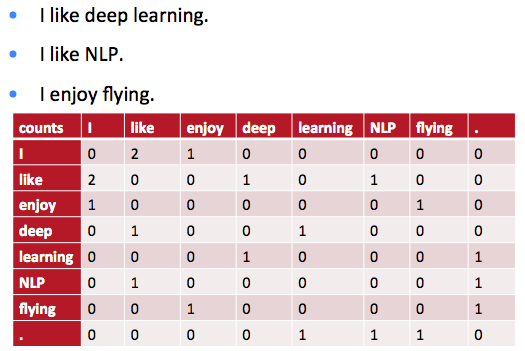
12. 词性标注

13. N-gram

传统的语言处理模型：

**1. Hyperspace Analogue to Language method (HAL)**

HAL (Lund & Burgess, 1996）方法可以用一个co-occurrence matrix, 表示任意两个词相关性。如图所示为一个window size=1的co-occurrence matrix结果：



这里window size 是指计算作用域。 比如window size=5就表示与一个词相邻5个词为作用域， weight随相邻词距离增大，从5到1递减。 根据co-occurrence matrix， 可得每个词有一个vector表示， 然后可以用Euclidean distance的倒数， 或 cosine， 或相关系数表示任意两个词的相似度。

但这样存在几个问题：

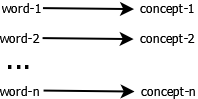
* 随着词汇增多， 矩阵大小增长， 耗存储
* 矩阵非常sparse的， 相应分类问题也需要考虑sparse模型。

于是我们想， 能不能降到低维形成一个dense的co-occurrence matrix **X**?

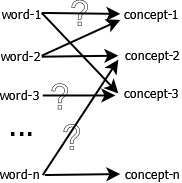
**潜在语义分析LSA**

一般流程：生成矩阵->调整内容权重->使用SVD(Singlar Value Decomposition)

潜在语义分析LSA（Latent Semantic Analysis ）也叫作潜在语义索引LSI（ Latent Semantic Indexing ） 顾名思义是通过分析文章（documents ）来挖掘文章的潜在意思或语义（concepts ）。如果每个单词都仅以着一个语义，同时每个语义仅仅由一个单词来表示，那么LSA将十分简单，即简单地将进行语义和单词间的映射。



不幸的是，LSA并没有这么简单。因为不同的单词可以表示同一个语义，或一个单词同时具有多个不同的意思，这些模糊歧义使语义的准确识别变得十分困难。

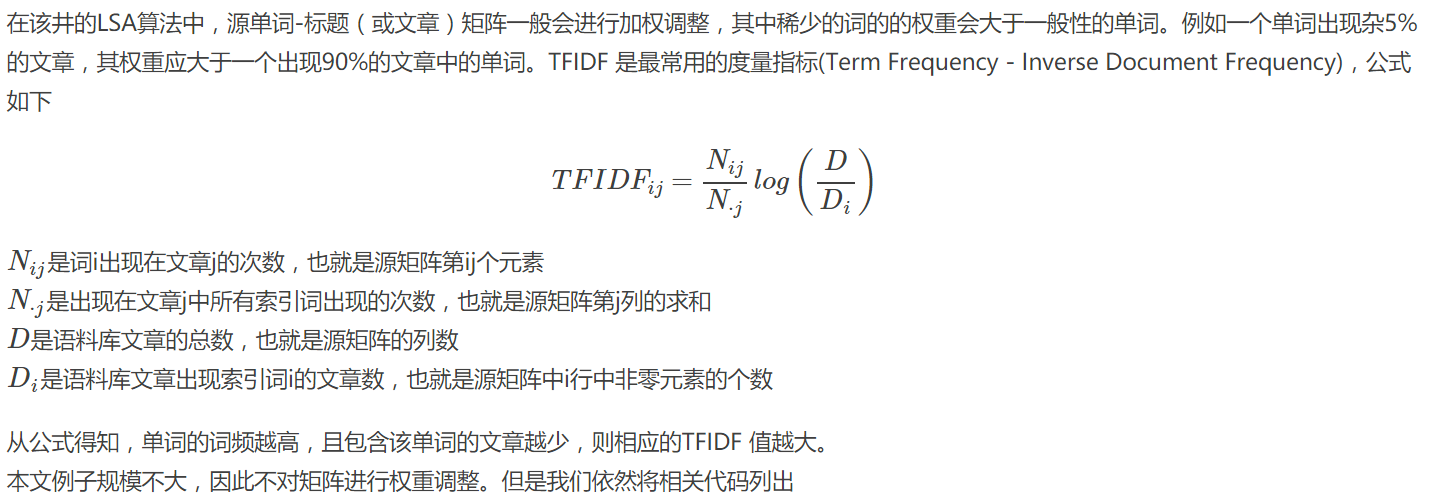


例如，bank 这个单词如果和mortgage, loans, rates 这些单词同时出现时，bank 很可能表示金融机构的意思。可是如果bank 这个单词和lures, casting, fish一起出现，那么很可能表示河岸的意思

LSA目的是解决如通过搜索词/关键词（search words）定位出相关文章。如何通过对比单词来定位文章是一个难点，因为我们正在要做的是对比单词背后的语义。潜在语义分析的基本原理是将文章和单词懂映射到语义空间（ “concept” space ）上，并在该空间进行对比分析。

由于作家在创作文章可以随意地选择各种单词来表达，因此不同的作家的词语选择风格都大不相同，表达的语义也因此变得模糊。这种单词选择的随机性必然将噪声的引入到“单词-语义关系”（word-concept relationship）。LSA能过滤掉一些噪声，同时能在语料库中找出一个最小的语义子集（ to find the smallest set of concepts that spans all the documents）。为了让问题变得课解，LSA引入了一些重要的假设

1. 文章通过”bags of words”的形式来表示，也就是说单词的出现顺数并不重要，而与单词在文中出现的次数相关
2. 语义通过一组最有可能同时出现的单词来表示。例如”leash”, “treat”, “obey” 常出现在关于 dog training的文章里面。
3. 每个单词假设只有一个意思，当然这个假设在遇到““banks””（既表示河岸也表示金融银行）这种情况当然不合适，但是这个假设将有助于简化问题难度。



LSA具有很多优势让其被广泛应用于各种领域

1. 首先文章和单词都映射到同一个语义空间。在该空间内即能对文章进行聚类也能对单词进行聚类。重要的是我们能通过这些聚类结果实现基于单词的文献检索，反之亦然。
2. 语义空间的维度明显明显少于源单词-文章矩阵。更重要的是这样经过特定方式组合而成维度包含源矩阵的大量信息，同时降低了噪声的影响。这些特性有助于后续其他算法的加工处理。
3. 最后，LSA 是一个全局最优化算法，其目标是寻找全局最优解而非局部最优解，因此它能求出基于局部求解算法得不到的全局信息。有时LSA会结合一些局部算法，如最近领域法，使得LSA性能得到进一步提升

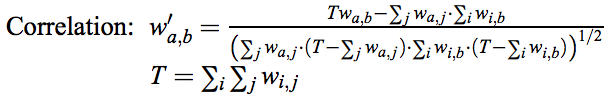
LSA依然存在一些缺陷，在我们使用的时候需要特别注意。

1. 首先LSA是假设服从高斯分布和2范数规范化的，因此它并非适合于所有场景。例如，单词在语料库中服从的是Poisson 分布而不是高斯分布
2. LSA不能有效处理一词多义问题。因为LSA的基本假设之一是单词只有一个词义
3. LSA的核心是SVD，而SVD的计算复杂度十分高并且难以更新新出现的文献。不过最近已经出现一些有效的方法用于解决SVD的基于文献更新问题。

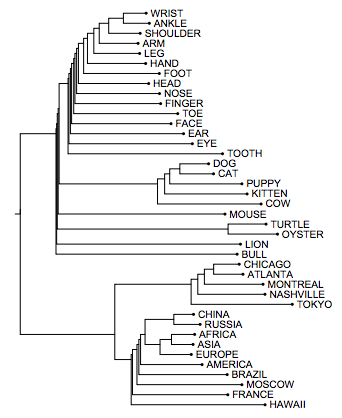
即使SVD存在上述缺陷，当LSA依然被广泛用于文献检索，文本分类，垃圾邮件过滤，语言识别，模式检索以及文章评估自动化等场景

COALS （Rohde et al., 2009）

在HAL上做了小改动， 将HAL所得co-occurrence matrix进行correlation normalization，



然后由于负相关的不可靠性， 将所有负相关置零得到新的co-occurrence matrix。 实验证明其数据清洗更好， 且满足高维稀疏性， 可进行快速SVD。 这是文章中的图，聚类效果还不错：



继上次分享了[经典统计语言模型](http://blog.csdn.net/abcjennifer/article/details/45419591)，最近公众号中有很多做NLP朋友问到了关于word2vec的相关内容， 本文就在这里整理一下做以分享。 本文分为

* 概括word2vec
* 相关工作
* 模型结构
* Count-based方法 vs. Directly predict

几部分，暂时没有加实验章节，但其实感觉word2vec一文中实验还是做了很多工作的，希望大家有空最好还是看一下~

概括word2vec

要解决的问题： 在神经网络中学习将word映射成连续（高维）向量， 其实就是个词语特征求取。

特点：   
1. 不同于之前的计算cooccurrence次数方法，减少计算量   
2. 高效   
3. 可以轻松将一个新句子/新词加入语料库

主要思想：神经网络语言模型可以用两步进行训练：1. 简单模型求取word vector; 在求取特征向量时，预测每个词周围的词作为cost 2. 在word vector之上搭建N-gram NNLM，以输出词语的概率为输出进行训练。

## 相关工作

在传统求取word的空间向量表征时， LSA 将词和文档映射到潜在语义空间，从而去除了原始向量空间中的一些“噪音”，但它无法保存词与词之间的linear regularities； LDA 是一个三层贝叶斯概率模型，包含词、主题和文档三层结构。文档到主题服从Dirichlet分布，主题到词服从多项式分布， 但是只要训练数据大了， 计算量就一下飚了。

基于神经网络的词语向量表征方法在[Y. Bengio, R. Ducharme, P. Vincent. A neural probabilistic language model, JMLR 2003]中就有提出， 名为NNLM， 它是一个前向网络， 同时学习词语表征和一个统计语言模型（后面具体讲）。

在Mikolov的硕士论文[1]和他在ICASSP 2009上发表的文章[2]中， 用一个单隐层网络训练词语表征， 然后将这个表征作为NNLM的输入进行训练。 Word2vec是训练词语表征工作的一个拓展。

## 模型结构

首先回顾NNLM，RNNLM，然后来看Word2Vec中提出的网络——CBOW，skip-gram Model。

1 . NNLM[3]

NNLM的目标是在一个NN里，求第t个词的概率， 即

NNLM_PROB

其中f是这个神经网络, 包括 input，projection， hidden和output。将其分解为两个映射：C和g，C是word到word vector的特征映射(通过一个|V|\*D的映射矩阵实现)，也称作look-up table， g是以word特征为输入，输出|V|个词语概率的映射：

这里写图片描述

如下图所示：

输入： n个之前的word（其实是他们的在词库V中的index）   
映射： 通过|V|\*D的矩阵C映射到D维   
隐层： 映射层连接大小为H的隐层   
输出： 输出层大小为|V|，表示|V|个词语的概率



用parameter个数度量网络复杂度， 则这个网络的复杂度为：

O=N∗D+N∗D∗H+H∗V

其中复杂度最高的部分为H\*V, 但通常可以通过hierarchical softmax或binary化词库编码将|V|降至log2V， 这样计算瓶颈就在于隐层N∗D∗H了。在word2vec中，为了避免隐层带来的高计算复杂度而去掉了隐层。

2 . RNNLM

RNN在语言模型上优于其他神经网络，因为不用像上面NNLM中的输入要定死前N个词的N。（具体RNN的结构我会在下篇中讲）简单地说， RNN就是一个隐层自我相连的网络， 隐层同时接收来自t时刻输入和t-1时刻的输出作为输入， 这使得RNN具有短期记忆能力， 所以RNNLM的复杂度为：

O=H∗H+H∗V

同样地，其中H∗V也可以降至log2V， 瓶颈就在于H∗H了。

由于复杂度最大的部分都在hidden layer, 而且我们的中级目标是提特征（而不是生成语言模型），文中就想能不能牺牲hidden layer的非线性部分， 从而高效训练。 这也是Word2vec中速度提升最多的部分。 这也就是一个Log linear model。所以本质上， word2vec并不是一个深度模型。文中提出了两种log linear model，如下面所述。

3 . Proposed Method 1 - Continuous Bag-of-Words(CBOW) Model

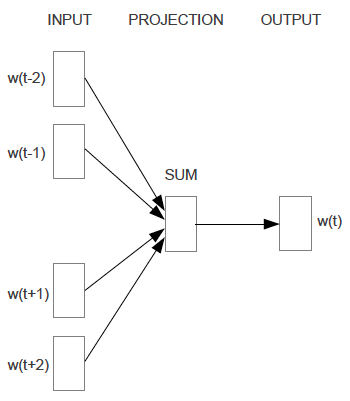
CBOW的网络结构和NNLM类似，变化：

1. CBOW去掉了NNLM的非线性部分
2. CBOW不考虑word之间的先后顺序， 一起放进bag，也就是在上面NNLM的projection层将映射后的结果求和/求平均（而非按照先后顺序连接起来）
3. 输入不止用了历史词语，还用了未来词语。 即， 用t-n+1…t-1,t+1,…t+n-1的word作为输入，目标是正确分类得到第t个word。   
   PS: 实验中得到的best n=4

CBOW的复杂度为：

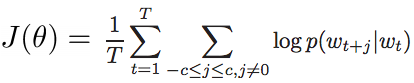
O=N∗D+D∗log2V

CBOW结构图：

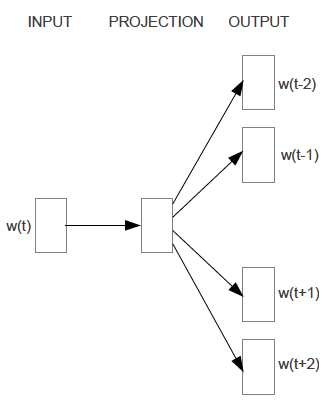


3 . Proposed Method 2 - Continuous Skip-gram Model

与CBOW相反，Continuous Skip-gram Model不利用上下文。 其输入为当前word，经过projection的特征提取去预测该word周围的c个词，其cost function为：



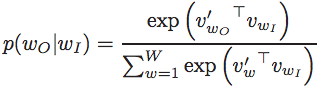
如下图所示。这里c增大有利于模型的完备性， 但过大的c可能造成很多无关词语相关联， 因此用随机采样方法，远的词少采， 近的多采。



比如定义最大周围距离为C，则对于每个词w(t)，就选择距离为R=range(1,C)， 选前后各R个词作为预测结果。   
所以，Continuous Skip-gram Model的复杂度为：

O=2C∗(D+D∗log2V)

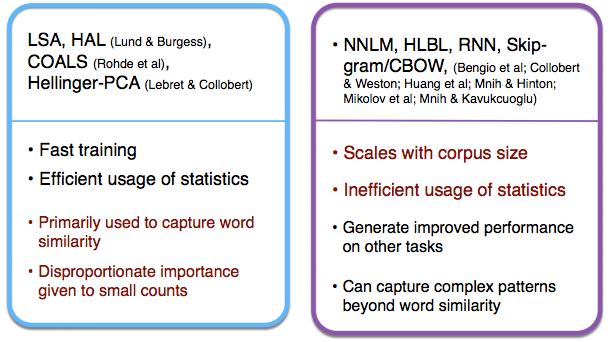
具体来说，最简单的情况下， P(wt+j|wt)的表达式可以为：



其中v和v’分别为输入和输出中的word特征向量。所以说， word2vec方法本质上是一个动态的逻辑回归。

## Count-based方法 vs. Directly predict

最后我们看一下之前我们讲过的几个[基于统计的传统语言模型](http://blog.csdn.net/abcjennifer/article/details/45419591)与word2vec这种直接预测的方法的比较：



作者：Scofield  
链接：https://www.zhihu.com/question/53354714/answer/155313446  
来源：知乎  
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

很好，正好可借此机会介绍**词向量、word2vec**以及**DeepNLP整套相关**的东西：

文章很长，是从CSDN上写好复制过来的，亦可直接跳到博客观看：

[深度学习（Deep Learning）·自然语言处理（NLP）·表示（Representation）](http://blog.csdn.net/scotfield_msn/article/details/69075227)

here we go.

*Indexing:*

**〇、序**

**一、DeepNLP的核心关键：语言表示（Representation）**

**二、NLP词的表示方法类型**

1、词的独热表示one-hot representation

2、词的分布式表示distributed representation

**三、NLP语言模型**

**四、词的分布式表示**

1. 基于矩阵的分布表示

2. 基于聚类的分布表示

3. 基于神经网络的分布表示，词嵌入（ word embedding）

**五、词嵌入（ word embedding）**

1、概念

2、理解

**六、神经网络语言模型与word2vec**

1、神经网络语言模型

2.word2vec与CBOW、Skip-gram

3.个人对word embedding的理解

**七、后言**

**References**

**〇、序**

之前一段时间，在结合[**深度学习**](http://lib.csdn.net/base/deeplearning)做NLP的时候一直有思考一些问题，其中有一个问题算是最核心一个：究竟深度网络是怎么做到让各种NLP任务解决地如何完美呢？到底我的数据在NN中发什么了什么呢？并且，不少的terms like： 词向量、word embedding、分布式表示、word2vec、glove等等，这一锅粥的名词术语分别代表什么，他们具体的关系是什么，他们是否处于平级关系？出于对知识结构追求完整梳理的强迫症的老毛病，于是不停地查资料、思考、keep revolving……然后就感觉有一点小进展了。想到，不如将个人对其的理解，无论对错，先拿出来跟peer分享下，或许能交换出更有意义的东西呢？

整篇文章的构架是按照属于概念在逻辑上的先后大小顺序，一层一层一级一级地往下剖析、比较、说明。

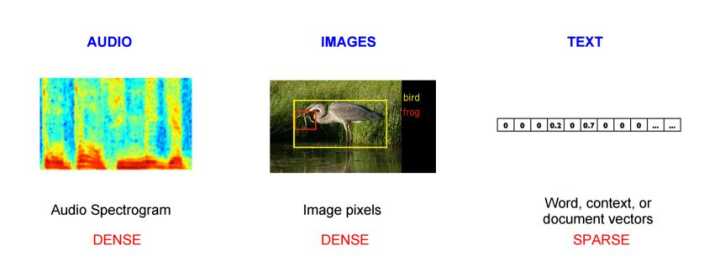
另外说明下，here整篇文字内容相对是比较入门，甚至有的点可能描述的不太客观正确，限于当前的认知水平……还请您海涵，希望您在评论中指正！

**一、DeepNLP的核心关键：语言表示（Representation）** 最近有一个新名词：Deep Learning + NLP = DeepNLP。当常规的[**机器学习**](http://lib.csdn.net/base/machinelearning)Machine Learning升级发展到了一定的阶段后，慢慢的被后起的深度学习Deep Learning夺势而去，并如火如荼地引领了一波新高潮，因为Deep Learning有machinelearning过而不及之处！那当Deep Learning进入NLP领域，自然是要横扫ACL一批paper才是。事实也是这样的。

先提下数据特征表示问题。数据表示是机器学习的核心问题，在过去的Machine Learning阶段，大量兴起特征工程，人工设计大量的特征解决数据的有效表示问题。而到了Deep Learning，想都别想，end-2-end，一步到位，hyper-parameter自动帮你选择寻找关键的特征参数。

那么，Deep Learning如何能在NLP中发挥出应有的real power呢？很明显，先不提如何设计出很强势的网络结构，不提如何在NLP中引入基于NN的解决例如情感分析、实体识别、机器翻译、文本生成这些高级任务，咱们首先得把语言表示这一关过了——如何让语言表示成为NN能够处理的数据类型。

我们看看图像和语音是怎么表示数据的：



在语音中，用音频频谱序列向量所构成的matrix作为前端输入喂给NN进行处理，good；在图像中，用图片的像素构成的matrix展平成vector后组成的vector序列喂给NN进行处理，good；那在自然语言处理中呢？噢你可能知道或者不知道，将每一个词用一个向量表示出来！想法是挺简单的，对，事实上就是这么简单，然而真有这么简单吗？可能没这么简单。

有人提到，图像、语音属于比较自然地低级数据表示形式，在图像和语音领域，最基本的数据是信号数据，我们可以通过一些距离度量，判断信号是否相似，在判断两幅图片是否相似时，只需通过观察图片本身就能给出回答。而语言作为人类在进化了几百万年所产生的一种高层的抽象的思维信息表达的工具，其具有高度抽象的特征，文本是符号数据，两个词只要字面不同，就难以刻画它们之间的联系，即使是“麦克风”和“话筒”这样的同义词，从字面上也难以看出这两者意思相同（语义鸿沟现象），可能并不是简单地一加一那么简单就能表示出来，而判断两个词是否相似时，还需要更多的背景知识才能做出回答。

那么据上是不是可以自信地下一个结论呢：如何有效地表示出语言句子是决定NN能发挥出强大拟合计算能力的关键前提！

**二、NLP词的表示方法类型**

接下来将按照上面的思路，引出各种词的表示方法。按照现今目前的发展，词的表示分为独热表示one-hot、分布式表示distributed。

**1、词的独热表示one-hot representation** NLP 中最直观，也是到目前为止最常用的词表示方法是 One-hot Representation，这种方法把每个词表示为一个很长的向量。这个向量的维度是词表大小，其中绝大多数元素为 0，只有一个维度的值为 1，这个维度就代表了当前的词。关于one-hot编码的资料很多，街货，这里简单举个栗子说明：

“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 ...]

每个词都是茫茫 0 海中的一个 1。这种 One-hot Representation 如果采用稀疏方式存储，会是非常的简洁：也就是给每个词分配一个数字 ID。比如刚才的例子中，话筒记为 3，麦克记为 8（假设从 0 开始记）。如果要编程实现的话，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**](http://lib.csdn.net/base/datastructure)已经很好地完成了 NLP 领域的各种主流任务。

现在我们分析他的不当处。1、向量的维度会随着句子的词的数量类型增大而增大；2、任意两个词之间都是孤立的，根本无法表示出在语义层面上词语词之间的相关信息，而这一点是致命的。

**2、词的分布式表示distributed representation**

传统的独热表示（ one-hot representation）仅仅将词符号化，不包含任何语义信息。如何将语义融入到词表示中？Harris 在 1954 年提出的分布假说（ distributional hypothesis）为这一设想提供了理论基础：上下文相似的词，其语义也相似。Firth 在 1957 年对分布假说进行了进一步阐述和明确：词的语义由其上下文决定（ a word is characterized by thecompany it keeps）。

到目前为止，基于分布假说的词表示方法，根据建模的不同，主要可以分为三类：基于矩阵的分布表示、基于聚类的分布表示和基于神经网络的分布表示。尽管这些不同的分布表示方法使用了不同的技术手段获取词表示，但由于这些方法均基于分布假说，它们的核心思想也都由两部分组成：一、选择一种方式描述上下文；二、选择一种模型刻画某个词（下文称“目标词”）与其上下文之间的关系。

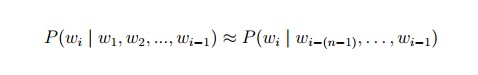
**三、NLP语言模型**

在详细介绍词的分布式表示之前，需要将NLP中的一个关键概念描述清楚：语言模型。语言模型包括文法语言模型和统计语言模型。一般我们指的是统计语言模型。之所以要将语言模型摆在词表示方法之前，是因为后面的表示方法马上要用到这一概念。

统计语言模型： 统计语言模型把语言（词的序列）看作一个随机事件，并赋予相应的概率来描述其属于某种语言集合的可能性。给定一个词汇集合 V，对于一个由 V 中的词构成的序列S = ⟨w1, · · · , wT ⟩ ∈ Vn，统计语言模型赋予这个序列一个概率P(S)，来衡量S 符合自然语言的语法和语义规则的置信度。

用一句简单的话说，就语言模型就是计算一个句子的概率大小的这种模型。有什么意义呢？一个句子的打分概率越高，越说明他是更合乎人说出来的自然句子。

就是这么简单。常见的统计语言模型有N元文法模型（N-gram Model），最常见的是unigram model、bigram model、trigram model等等。形式化讲，统计语言模型的作用是为一个长度为 m 的字符串确定一个概率分布 P(w1; w2; :::; wm)，表示其存在的可能性，其中 w1 到 wm 依次表示这段文本中的各个词。一般在实际求解过程中，通常采用下式计算其概率值：



同时通过这些方法均也可以保留住一定的词序信息，这样就能把一个词的上下文信息capture住。

具体的语言模型详情属于街货，详细请自行搜索。

**四、词的分布式表示**

**1. 基于矩阵的分布表示** 基于矩阵的分布表示通常又称为分布语义模型，在这种表示下，矩阵中的一行，就成为了对应词的表示，这种表示描述了该词的上下文的分布。由于分布假说认为上下文相似的词，其语义也相似，因此在这种表示下，两个词的语义相似度可以直接转化为两个向量的空间距离。

常见到的Global Vector 模型（ GloVe模型）是一种对“词-词”矩阵进行分解从而得到词表示的方法，属于基于矩阵的分布表示。

**2. 基于聚类的分布表示** 基于聚类的分布表示我也还不是太清楚，所以就不做具体描述。

**3. 基于神经网络的分布表示，词嵌入（ word embedding）**

**基于神经网络的分布表示一般称为词向量、词嵌入（ word embedding）或分布式表示（ distributed representation）**。这正是我们的主角today。

神经网络词向量表示技术通过神经网络技术对上下文，以及上下文与目标词之间的关系进行建模。由于神经网络较为灵活，这类方法的最大优势在于可以表示复杂的上下文。在前面基于矩阵的分布表示方法中，最常用的上下文是词。如果使用包含词序信息的 n-gram 作为上下文，当 n 增加时， n-gram 的总数会呈指数级增长，此时会遇到维数灾难问题。而神经网络在表示 n-gram 时，可以通过一些组合方式对 n 个词进行组合，参数个数仅以线性速度增长。有了这一优势，神经网络模型可以对更复杂的上下文进行建模，在词向量中包含更丰富的语义信息。

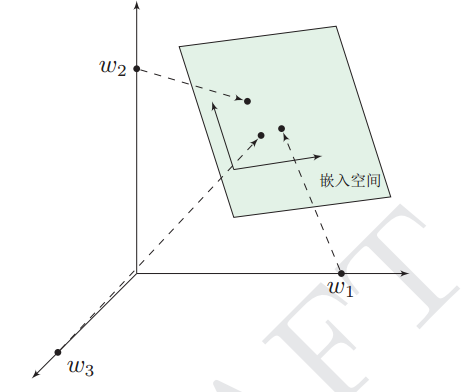
**五、词嵌入（ word embedding）**

**1、概念** 基于神经网络的分布表示又称为词向量、词嵌入，神经网络词向量模型与其它分布表示方法一样，均基于分布假说，核心依然是上下文的表示以及上下文与目标词之间的关系的建模。

前面提到过，为了选择一种模型刻画某个词（下文称“目标词”）与其上下文之间的关系，我们需要在词向量中capture到一个词的上下文信息。同时，上面我们恰巧提到了统计语言模型正好具有捕捉上下文信息的能力。那么构建上下文与目标词之间的关系，最自然的一种思路就是使用语言模型。从历史上看，早期的词向量只是神经网络语言模型的副产品。

2001年， Bengio 等人正式提出神经网络语言模型（ Neural Network Language Model ，NNLM），该模型在学习语言模型的同时，也得到了词向量。所以请注意一点：**词向量可以认为是神经网络训练语言模型的副产品**。

**2、理解** 前面提过，one-hot表示法具有维度过大的缺点，那么现在将vector做一些改进：1、将vector每一个元素由整形改为浮点型，变为整个实数范围的表示；2、将原来稀疏的巨大维度压缩嵌入到一个更小维度的空间。如图示：



这也是词向量又名词嵌入的缘由了。

**六、神经网络语言模型与word2vec**

好了，到目前为止我们已经对的分布式表示以及词嵌入的概念的层级关系有了个理性的认识了，那这跟word2vec有什么联系？

**1、神经网络语言模型**

上面说，通过神经网络训练语言模型可以得到词向量，那么，究竟有哪些类型的神经网络语言模型呢？个人所知，大致有这么些个：

a) Neural Network Language Model ，NNLM b) Log-Bilinear Language Model， LBL c) Recurrent Neural Network based Language Model，RNNLM d) Collobert 和 Weston 在2008 年提出的 C&W 模型 e) Mikolov 等人提出了 CBOW（ Continuous Bagof-Words）和 Skip-gram 模型

到这，估计有人看到了两个熟悉的term：CBOW、skip-gram，有看过word2vec的同学应该对此有所了解。我们继续。

**2.word2vec与CBOW、Skip-gram**

现在我们正式引出最火热的另一个term：word2vec。

上面提到的5个神经网络语言模型，只是个在逻辑概念上的东西，那么具体我们得通过设计将其实现出来，而实现CBOW（ Continuous Bagof-Words）和 Skip-gram 语言模型的工具正是well-known word2vec！另外，C&W 模型的实现工具是SENNA。

所以说，分布式词向量并不是word2vec的作者发明的，他只是提出了一种更快更好的方式来训练语言模型罢了。分别是：连续词袋模型Continous Bag of Words Model(CBOW)和Skip-Gram Model，这两种都是可以训练出词向量的方法，再具体代码操作中可以只选择其一，不过据论文说CBOW要更快一些。

顺便说说这两个语言模型。统计语言模型statistical language model就是给你几个词，在这几个词出现的前提下来计算某个词出现的（事后）概率。CBOW也是统计语言模型的一种，顾名思义就是根据某个词前面的C个词或者前后C个连续的词，来计算某个词出现的概率。Skip-Gram Model相反，是根据某个词，然后分别计算它前后出现某几个词的各个概率。

以“我爱北京天安门”这句话为例。假设我们现在关注的词是“爱”，C＝2时它的上下文分别是“我”，“北京天安门”。CBOW模型就是把“我” “北京天安门” 的one hot表示方式作为输入，也就是C个1xV的向量，分别跟同一个VxN的大小的系数矩阵W1相乘得到C个1xN的隐藏层hidden layer，然后C个取平均所以只算一个隐藏层。这个过程也被称为线性激活函数(这也算激活函数？分明就是没有激活函数了)。然后再跟另一个NxV大小的系数矩阵W2相乘得到1xV的输出层，这个输出层每个元素代表的就是词库里每个词的事后概率。输出层需要跟ground truth也就是“爱”的one hot形式做比较计算loss。这里需要注意的就是V通常是一个很大的数比如几百万，计算起来相当费时间，除了“爱”那个位置的元素肯定要算在loss里面，word2vec就用基于huffman编码的Hierarchical softmax筛选掉了一部分不可能的词，然后又用nagetive samping再去掉了一些负样本的词所以时间复杂度就从O(V)变成了O(logV)。Skip gram训练过程类似，只不过输入输出刚好相反。

补充下，Word embedding的训练方法大致可以分为两类：一类是无监督或弱监督的预训练；一类是端对端（end to end）的有监督训练。无监督或弱监督的预训练以word2vec和auto-encoder为代表。这一类模型的特点是，不需要大量的人工标记样本就可以得到质量还不错的embedding向量。不过因为缺少了任务导向，可能和我们要解决的问题还有一定的距离。因此，我们往往会在得到预训练的embedding向量后，用少量人工标注的样本去fine-tune整个模型。

相比之下，端对端的有监督模型在最近几年里越来越受到人们的关注。与无监督模型相比，端对端的模型在结构上往往更加复杂。同时，也因为有着明确的任务导向，端对端模型学习到的embedding向量也往往更加准确。例如，通过一个embedding层和若干个卷积层连接而成的深度神经网络以实现对句子的情感分类，可以学习到语义更丰富的词向量表达。

**3.个人对word embedding的理解**

现在，词向量既能够降低维度，又能够capture到当前词在本句子中上下文的信息（表现为前后距离关系），那么我们对其用来表示语言句子词语作为NN的输入是非常自信与满意的。

另外一点很实用的建议，在你做某一项具体的NLP任务时如你要用到词向量，那么我建议你：要么**1、选择使用别人训练好的词向量，注意，得使用相同语料内容领域的词向量；要么2、自己训练自己的词向量。**我建议是前者，因为……坑太多了。

对word2vec的评价指标：当前绝大部分工作（比如以各种方式改进word embedding）都是依赖wordsim353等词汇相似性数据集进行相关性度量，直接用马氏距离或者cos计算个分类器，看对近义词的分类效果。并以之作为评价word embedding质量的标准。然而，这种基于similarity的评价方式对训练数据大小、领域、来源以及词表的选择非常敏感。而且数据集太小，往往并不能充分说明问题

对于上面的这个评价问题，我们也做了不少的尝试：

 尝试了word analogy的方法，收集了一堆类似“东来顺-海底捞+梅州东坡=便宜坊”这种case，有点意思但没啥用，因为商品之间的关系不像语言中那么明确（手动皱眉）

 结合业务场景，尝试了用KNN的信息（价格、位置、品类等）预测当前商品的信息，做下来准确率还蛮好看的，和线上效果也基本一致。

 别的还试过tensorflow做embedding的可视化，不过这个是用来写ppt（逃）

 公司里当然还是线上效果最有说服力。