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！

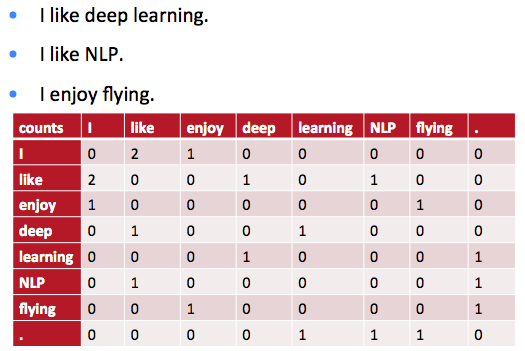
12. 词性标注

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：

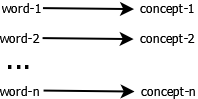
* 随着词汇增多， 矩阵大小增长， 耗存储
* 矩阵非常sparse的， 相应分类问题也需要考虑sparse模型。

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

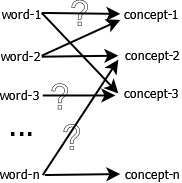
**潜在语义分析LSA**

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。

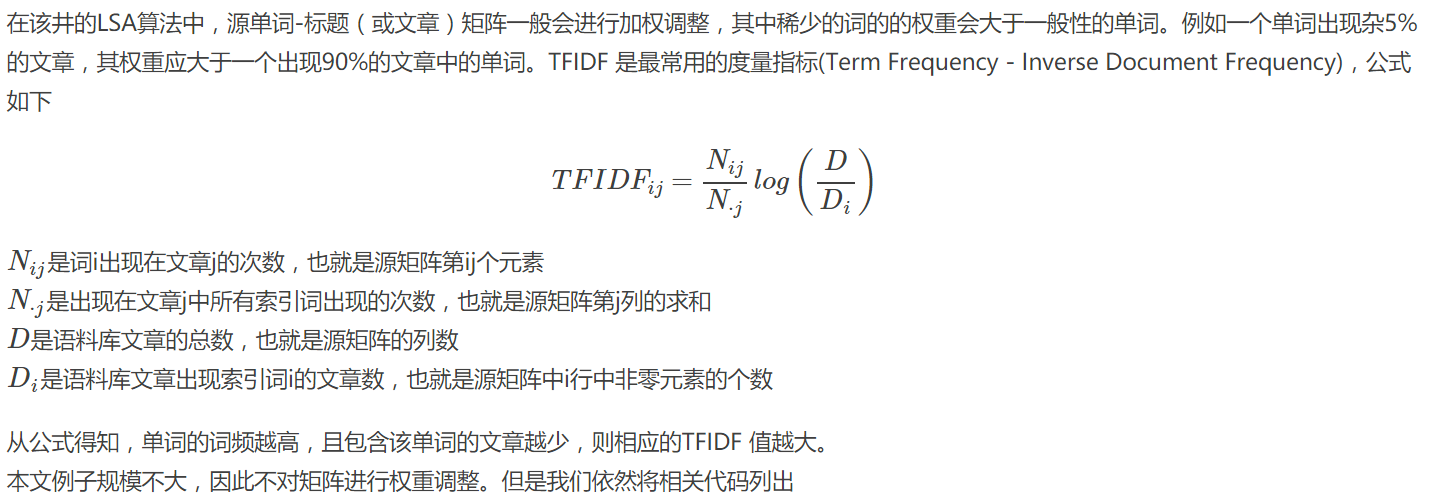


例如，bank 这个单词如果和mortgage, loans, rates 这些单词同时出现时，bank 很可能表示金融机构的意思。可是如果bank 这个单词和lures, casting, fish一起出现，那么很可能表示河岸的意思

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

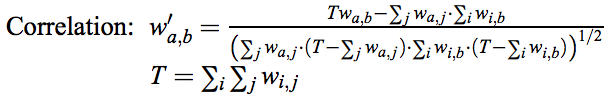
LSA依然存在一些缺陷，在我们使用的时候需要特别注意。

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。

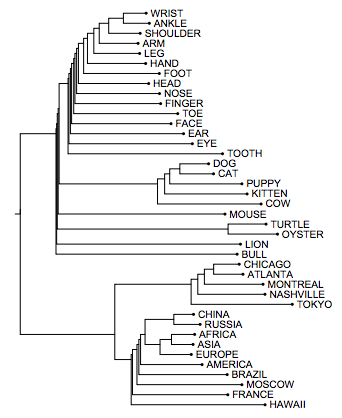
即使SVD存在上述缺陷，当LSA依然被广泛用于文献检索，文本分类，垃圾邮件过滤，语言识别，模式检索以及文章评估自动化等场景

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

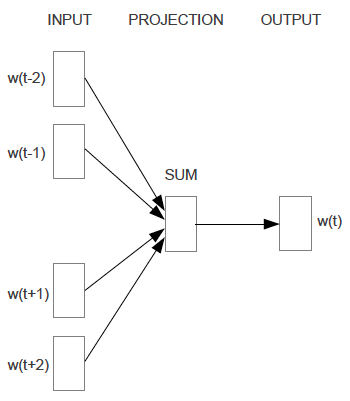
CBOW的网络结构和NNLM类似，变化：

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：

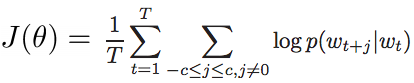
O=N∗D+D∗log2V

CBOW结构图：

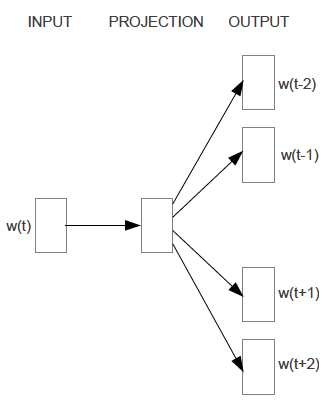


3 . Proposed Method 2 - Continuous Skip-gram Model

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：



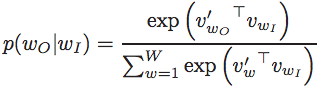
如下图所示。这里c增大有利于模型的完备性， 但过大的c可能造成很多无关词语相关联， 因此用随机采样方法，远的词少采， 近的多采。



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：

