LSA

2015年7月27日 星期一 下午10:34

一,简介
LSA,(latent semantic analyse) ,潜在语义分析.,基于统计,刻画了词-文档之间的潜在的关系, 进而由词-文档 推广到了词-词之间的相似度.
之所以说"潜在",有别于 最土的方式: 构建词-文档的共现矩阵,用0/1表示是否出现,他在这里是在最土的方式上改进，做了降维，降维强调了重要的特征/concept/矩阵变换的方向，消除了噪声的方向，提高了信噪比？。 (~~LSA对没有共现的词？？,也能有个潜在的语义上的估计~~？？？)
是PLSA的前身

二，引子
构建词-文档的关系,一个很土的方式就是 构建词-文档的共现矩阵,~~用0/1表示是否出现~~,用每个word在每个document中出现的次数表示，但是这样对于噪音等容错效果差,矩阵稀疏,维度高,有很多问题 (~~LSA 可以对没有共现的词,也能有个潜在的语义上的估计~~？) 好处见参考资料1 中例子

三.算法过程
大致:
1. 分析文档集合，建立Term-Document矩阵。
2. 对Term-Document矩阵进行奇异值分解。
3. 对SVD分解后的矩阵进行降维，也就是奇异值分解一节所提到的低阶近似。
4. 使用降维后的矩阵构建潜在语义空间，或重建Term-Document矩阵。

第一次看的理解：
第一步, Term-Document矩阵,就是建立很土的那矩阵 M (n*m)
第二步,用SVD进行对 Term-Document矩阵进行 分解 ,过程例子见参考资料1
    但是有个图错了 SVD要求 U 和 V 都是方阵 中间的 ∑ 的维度可以不用填满
    就是简单的分成三个矩阵 ~~其实没什么物理意义 唯一的意义就是∑的矩阵的值 是奇异值~~
    M (n*m) = U (n*n) * ∑ (n*m) * V (m*m) 其中 ∑的n*m 不用填满 比如目前有k维非零
第三步，实现降维(低阶近似)
    方法是: 对奇异值取top K ??? 还是必须利用F-范数???
    当然U 和V 的对应维 也要跟着∑ 一起拿出来
    M (n*m) = U (n*n) * ∑ (n*m) * V (m*m) 其中 ∑ 从之前的k维 变为d维非零 (d<k)
第四步，使用降维后的矩阵构建潜在语义空间，或重建Term-Document矩阵

第四步，使用降维后的矩阵构建潜在语义空间，或重建Term-Document矩阵。

可以把已经降过维的乘在一起

Md (n*m) = Ud (n*d) * ∑d (d*d) * Vd(d*m) 这个形式和 word2vec用SVD解读的paper中的表示是一致的

Ud和 Vd 是U和V 按照∑ 的标准进行降维, 也就是第三步中∑ 是0的行和列 在U V 中被去掉 去掉之后的矩阵就是Ud 和Vd

Md 虽然维度还是这么高 但是我们使用时不会用 Md ,会用Ud 和Vd的

第二次看：

第一次的见解中有一些可取的地方，但是看了英文博客，有了更深入的理解

英文博客大致内容翻译：

1、计算count matrix （计数矩阵），这个就是word-document的矩阵；但是这个基于一些假设，

A 没有歧义：一个词只有一个concept，一个concept对应一个词（因为这里直接word-document，说明他认为word能够完全表示semantic）

B 词袋模型：基于bagofword，不考虑词的顺序，只考虑出现次数

C 基于共现：相似的词总是一起出现（这样从统计的角度，得出词的相似度，才有意义）

例子：建立word-title的矩阵，注意出现两次的次，是2

Index Words	Titles								
	T1	T2	T3	T4	T5	T6	T7	T8	T9
book			1	1					
dads						1			1
dummies		1						1	
estate							1		1
guide	1					1			
investing	1	1	1	1	1	1	1	1	1
market	1		1						
real							1		1
rich						2			1
stock	1		1					1	
value				1	1				

2、可以考虑TFIDF，用TFIDF的值，代替矩阵内部的0/1/....这些

review一下：tfidf = i词在j文档中的频次/j文档中所有词数 * log(总文档数/包含i词的文档数)

3、SVD分解

其实就是对1或2中的矩阵做一个降维。

SVD的目的是对term-document的矩阵进行降维，降维的数学操作先不管，我们看看有什么物理意义。

我们做SVD分解M = U ∑ V 之后，一个矩阵成了三个矩阵，非要给U和V找一种物理含义的话，吴军有一些解释

过LSI也是一个严重依赖于SVD的算法，之前吴军老师在矩阵计算与文本处理中的分类问题中谈到：

"三个矩阵有非常清楚的物理含义。第一个矩阵X中的每一行表示意思相关的一类词，其中的每个非零元素表示这类词中每个词的重要性（或者说相关性），数值越大越相关。最后一个矩阵Y中的每一列表示同一主题一类文章，其中每个元素表示这类文章中每篇文章的相关性。中间的矩阵则表示类词和文章雷之间的相关性。

因此，我们只要对关联矩阵A进行一次奇异值分解，w 我们就可以同时完成了近义词分类和文章的分类。（同时得到每类文章和每类词的相关性）。"

在word和doc之间如果加一层的话，那就应该是语义/类/concept/语义空间这些，

如果整体上从数学的角度思考，U和V的内部都是一组正交向量，是互相之间线性无关的，可以认为是n维空间的n个坐标轴，这样的话，其实我们做的就是，在这n维空间中，由于word-doc的sparse特性，一大部分维度(坐标轴)是没有用的，反而是噪声，这个应该去掉，只留下几个重要的独立成分。U和V都是这样的

最后U(n*n) V(m*m) 降维变成了 U(n*r) V(r*m)

SVD分解可以看做是，重点保留有用的方向，去除一些可能是噪声的方向。

四,使用

1, Md指定没法用了

2, sim

由于Md 的每个行向量代表了n个词中每个词包含的潜在的和哪些文档匹配 (下称为 主题), 所以,想计算第i词语第j词之间的相似度,只需要第i行与第j行 的行向量之间点乘就行了

如果用一个矩阵来表示 那就说 W = X * X^T W的第i行 j列的元素wij 就代表 第i行与第j行 的行向量之间内积 也就是 第i词与第j词之间的相似度

对 W = X * X^T 进行化简，根据SVD的性质 $MM^* = U\Sigma V^* V\Sigma^* U^* = U(\Sigma\Sigma^*)U^*$ $M^*M = V\Sigma^* U^* U\Sigma V^* = V(\Sigma^*\Sigma)V^*$

W = U ∑ (U ∑)^T = U * ∑^2 * U^T

想计算文档-文档之间的相似度 同理 W = V * ∑^2 * V^T

使用的方式 就是用这个式子计算词-词,文档-文档之间的相似度

3, embedding

见参考资料2 那里面直接用 U * ∑ 作为了embedding 很赞!!

第二次看参考资料：

简单易懂的中文博客

http://www.cnblogs.com/LeftNotEasy/archive/2011/01/19/svd-and-applications.html

英文博客很靠谱 http://www.puffinwarellc.com/index.php/news-and-articles/articles/33-latent-semantic-analysis-tutorial.html

第一次看参考资料:

1, http://blog.csdn.net/wangran51/article/details/7408406 基础 入门级 SVD时有个图错了

2. https://levyomer.files.wordpress.com/2014/09/neural-word-embeddings-as-

implicit-matrix-factorization.pdf word2vec用SVD解读的paper过程和LSA基本一致

3, plsa的paper http://cs.brown.edu/~th/papers/Hofmann-UAI99.pdf 没看

4, http://blog.sina.com.cn/s/blog_62a9902f0101cjl3.html 没细看

5. http://en.wikipedia.org/wiki/Latent_semantic_analysis 没细看