

CSDN

博客 (//blog.csdn.net/?ref=toolbar) 学院 (//edu.csdn.net/?ref=toolbar) 下载 (//download.csdn.net/?ref=toolbar) GitChat (//gitbook.cn/?ref=csdn) 更多 ▾

weixin\_3506... (//my.csdn.net/?ref=toolbar) (//write.blog.csdn.net/postedit?ref=toolbar)source=csdnblog

itgeeks (http://blog.csdn....) + 关注

码云

未开通 (https://gitee.com/)

未开通 (https://gitee.com/)

gensim做主题模型

原创 2013年12月24日 15:28:00 标签：python (http://so.csdn.net/so/search/s.do?q=python&t=blog)

17212

作为python的一个库，gensim给了文本主题模型足够的方便，像他自己的介绍一样，topic modelling for humans

具体的tutorial可以参看他的官方网页，当然是全英文的，  
<http://radimrehurek.com/gensim/tutorial.html>

由于这个链接打开速度太慢太慢，我决定写个中文总结：（文章参考了52nlp的博客，参看<http://www.52nlp.cn>）

安装就不用说了，在ubuntu环境下，sudo easy\_install gensim即可

首先，引用gensim包，gensim包中引用corpora,models, similarities，分别做语料库建立，模型库和相似度比较库，后面可以看到例子  
from gensim import corpora, models, similarities

我调用了结巴分词做中文处理，所以同样  
import jieba

手工写个文本列表  
sentences = ["我喜欢吃土豆","土豆是个百搭的东西","我不喜欢今天雾霾的北京"]

用结巴分词后待用，因为gensim包做主题模型，在意的是语料库，所以，中文英文，one-term，two-term都是无所谓的，如果有已经生成好的语料库，那么可以考虑直接跳到建模环节

官方提供的语料库范例是这样的：

```
corpus = [[(0, 1.0), (1, 1.0), (2, 1.0)],
>>> [(2, 1.0), (3, 1.0), (4, 1.0), (5, 1.0), (6, 1.0), (8, 1.0)],
>>> [(1, 1.0), (3, 1.0), (4, 1.0), (7, 1.0)],
>>> [(0, 1.0), (4, 2.0), (7, 1.0)],
>>> [(3, 1.0), (5, 1.0), (6, 1.0)],
>>> [(9, 1.0)],
>>> [(9, 1.0), (10, 1.0)],
>>> [(9, 1.0), (10, 1.0), (11, 1.0)],
>>> [(8, 1.0), (10, 1.0), (11, 1.0)]]
```

他的最新文章

更多文章 (http://blog.csdn.net/whzhcahzhxh)

python操作redis (http://blog.csdn.net/whzhcahzhxh/article/details/41211213)

[算法9]shuffle算法 (http://blog.csdn.net/whzhcahzhxh/article/details/38895615)

mac osx系统g++编译c++ (http://blog.csdn.net/whzhcahzhxh/article/details/38702117)

[算法8]union find算法 (http://blog.csdn.net/whzhcahzhxh/article/details/38094761)

[算法7]page rank算法 (http://blog.csdn.net/whzhcahzhxh/article/details/38065819)

相关推荐

gensim文本主题模型推荐 (http://blog.csdn.net/fenixshichengzhang/article/details/51746959)

nlTK在python中的安装,以及nlTK的数据库 (http://blog.csdn.net/elikai/article/details/46848671)

基于gensim的文本主题模型(LDA)分析 (http://download.csdn.net/download/u010297828/9391608)  
380

主题模型TopicModel：通过gensim实现LDA (http://blog.csdn.net/pipisorry/article/details/46447561)



每个中括号代表一句话，用逗号隔开，（0，1.0）代表词典中编号为0的词出现了一次，以此类推，很好理解

回到过程中来，将范例的语句分词  
words=[]  
for doc in sentences:  
    words.append(list(jieba.cut(doc)))  
print words  
输出：  
[[u'u6211', u'u559c\u6b22', u'u5403', u'u571f\u8c46'], [u'u571f\u8c46', u'u662f, u'u4e2a', u'u767e', u'u642d', u'u7684', u'u4e1c\u897f'], [u'u6211', u'u4e0d', u'u559c\u6b22', u'u4ecalu5929', u'u96fe', u'u973e', u'u7684', u'u5317\u4eac']]  
此时输出的格式为unicode，不影响后期运算，因此我保留不变，如果想看分词结果可以用循环输出jieba分词结果

得到的分词结果构造词典  
dic = corpora.Dictionary(words)  
print dic  
print dic.token2id  
输出：  
Dictionary(15 unique tokens)  
{'x5x8cx97xexbalxac': 12, 'x6x90xad': 6, 'x7x9alx84': 9, 'x5x96x9cxex6xacxa2': 1, 'x4xb8x8d': 10, 'x4xb8x9cxex8xa5xbf': 4, 'x5x9c9fxex8xb1x86': 2, 'x9x9cxb': 14, 'x6x98xaf': 7, 'x4xb8xaa': 5, 'x9x9bxb': 13, 'x7x99xb': 8, 'x4xbbx8axex5xa4xa9': 11, 'x6x88x91': 3, 'x5x90x83': 0}  
可以看到各个词或词组在字典中的编号

为了方便看，我给了个循环输出：  
for word,index in dic.token2id.iteritems():  
    print word +" 编号为:"+ str(index)  
输出：  
北京 编号为:12  
搭 编号为:6  
的 编号为:9  
喜欢 编号为:1  
不 编号为:10  
东西 编号为:4  
土豆 编号为:2  
霾 编号为:14  
是 编号为:7  
个 编号为:5  
雾 编号为:13  
百 编号为:8  
今天 编号为:11  
我 编号为:3  
吃 编号为:0  
为什么是乱序，我也不知道

广告

在线课程



(http://www.baidu.com/cb.php?c=lgF\_pyfqhHmknjmsnjD0iZ0qntK9ujYzP1mznWR10Aw-



SHR1rjfkN100T1YkuWmvmvmtzPjFbuHT4uH0v0AwY5HDdnHnYnjT4PW00lgF\_5y9YiZ0lQzq-



(http://www.baidu.com/cb.php?c=lgF\_pyfqhHmknjmsnj0iZ0qntK9ujYzP1mznWR10Aw-



SHR1rjfkN100T1YkuWmvmvmtzPjFbuHT4uH0v0AwY5HDdnHnYnjT4PW00lgF\_5y9YiZ0lQzq-



SHR1rjfkN100T1YkuWmvmvmtzPjFbuHT4uH0v0AwY5HDdnHnYnjT4PW00lgF\_5y9YiZ0lQzq-



(http://www.baidu.com/cb.php?c=lgF\_pyfqhHmknjmsnj0iZ0qntK9ujYzP1mznWR10Aw-



SHR1rjfkN100T1YkuWmvmvmtzPjFbuHT4uH0v0AwY5HDdnHnYnjT4PW00lgF\_5y9YiZ0lQzq-



SHR1rjfkN100T1YkuWmvmvmtzPjFbuHT4uH0v0AwY5HDdnHnYnjT4PW00lgF\_5y9YiZ0lQzq-



SHR1rjfkN100T1YkuWmvmvmtzPjFbuHT4uH0v0AwY5HDdnHnYnjT4PW00lgF\_5y9YiZ0lQzq-



SHR1rjfkN100T1YkuWmvmvmtzPjFbuHT4uH0v0AwY5HDdnHnYnjT4PW00lgF\_5y9YiZ0lQzq-



SHR1rjfkN100T1YkuWmvmvmtzPjFbuHT4uH0v0AwY5HDdnHnYnjT4PW00lgF\_5y9YiZ0lQzq-



SHR1rjfkN100T1YkuWmvmvmtzPjFbuHT4uH0v0AwY5HDdnHnYnjT4PW00lgF\_5y9YiZ0lQzq-



SHR1rjfkN100T1YkuWmvmvmtzPjFbuHT4uH0v0AwY5HDdnHnYnjT4PW00lgF\_5y9YiZ0lQzq-



SHR1rjfkN100T1YkuWmvmvmtzPjFbuHT4uH0v0AwY5HDdnHnYnjT4PW00lgF\_5y9YiZ0lQzq-



SHR1rjfkN100T1YkuWmvmvmtzPjFbuHT4uH0v0AwY5HDdnHnYnjT4PW00lgF\_5y9YiZ0lQzq-

词典生成好之后，就开始生成语料库了

```
corpus = [dic.doc2bow(text) for text in words]
```

```
print corpus
```

输出：

```
[[[0, 1), (1, 1), (2, 1), (3, 1)], [(2, 1), (4, 1), (5, 1), (6, 1), (7, 1), (8, 1), (9, 1)], [(1, 1), (3, 1), (9, 1), (10, 1), (11, 1), (12, 1), (13, 1), (14, 1)]]
```

语料库的官方描述写的是向量空间模型格式的语料库，*Corpus* is simply an object which, when iterated over, returns its documents represented as sparse vectors.

官方说明给了一个tips：

In this example, the whole corpus is stored in memory, as a Python list. However, the corpus interface only dictates that a corpus must support iteration over its constituent documents. For very large corpora, it is advantageous to keep the corpus on disk, and access its documents sequentially, one at a time. All the operations and transformations are implemented in such a way that makes them independent of the size of the corpus, memory-wise.

大概意思就是说范例中的语料库是存在内存中的一个列表格式，但是接口对语料库的要求只是，支持在构建文本文件中可循环即可，因此对于很大的语料库，最好还是存在硬盘上，然后依次访问文件。这样可以不用考虑语料库的size，也避免了内存占用太多

此时，得到了语料库，接下来做一个TF-IDF变换

可以理解成 将用词频向量表示一句话 变换成为用 词的重要性向量表示一句话

（TF-IDF变换：评估一字词对于一个文件集或一个语料库中的其中一份文件的重要程度。字词的重要性随着它在文件中出现的次数成正比增加，但同时会随着它在语料库（<http://baike.baidu.com/view/686705.htm>）中出现的频率成反比下降。）

```
tfidf = models.TfidfModel(corpus)
```

```
vec = [(0, 1), (4, 1)]
```

```
print tfidf[vec]
```

```
corpus_tfidf = tfidf[corpus]
```

```
for doc in corpus_tfidf:
```

```
    print doc
```

输出：

```
[(0, 0.7071067811865475), (4, 0.7071067811865475)]
[(0, 0.8425587958192721), (1, 0.3109633824035548), (2, 0.3109633824035548), (3, 0.3109633824035548)]
[(2, 0.16073253746956623), (4, 0.4355066251613605), (5, 0.4355066251613605), (6, 0.4355066251613605), (7, 0.4355066251613605), (8, 0.4355066251613605), (9, 0.16073253746956623)]
[(1, 0.1586956620869655), (3, 0.1586956620869655), (9, 0.1586956620869655), (10, 0.42998768831312806), (11, 0.42998768831312806), (12, 0.42998768831312806), (13, 0.42998768831312806), (14, 0.42998768831312806)]
```

vec是查询文本向量，比较vec和训练中的三句话相似度

```
index = similarities.SparseMatrixSimilarity(tfidf[corpus], num_features=14)
```

```
sims = index[tfidf[vec]]
```

```
print list(enumerate(sims))
```

输出：

2853

R语言解决MongoDB中文编码问题 (<http://blog.csdn.net/whzhcahxh/article/details/17222303>)

2853



4



```
[(0, 0.59577906), (1, 0.30794966), (2, 0.0)]
```

表示和第1句话相似度为59.578%，和第二句话的相似度位30.79%，第三句没有相似度，我们看看vec这句话是什么：0为吃，4为东西，所以vec这句话可以是["吃东西"]或者["东西吃"]而第一句话"我喜欢吃土豆"."土豆是个百搭的东西"明显有相似度，而第三句话"我不喜欢今天雾霾的北京"，相似度几乎为0，至于为什么第一句比第二句更相似，就需要考虑Tfidf document representation和cosine similarity measure了

回到tfidf转换，接着训练LSI模型，假定三句话属于2个主题，

```
lsi = models.LsiModel(corpus_tfidf, id2word=dic, num_topics=2)
```

```
lsiout=lsi.print_topics(2)
```

```
print lsiout[0]
```

```
print lsiout[1]
```

输出：

```
0.532*吃 + 0.290*喜欢 + 0.290*我 + 0.258*土豆 + 0.253*霾 + 0.253*雾 + 0.253*北京 + 0.253*今天 + 0.253*不 + 0.166*东西
0.393*百 + 0.393*搭 + 0.393*东西 + 0.393*是 + 0.393*个 + -0.184*霾 + -0.184*雾 + -0.184*北京 + -0.184*今天 + -0.184*不
```

这就是基于SVD建立的两个主题模型内容

将文章投影到主题空间中

```
corpus_lsi = lsi[corpus_tfidf]
```

```
for doc in corpus_lsi:
```

```
    print doc
```

输出：

```
[(0, -0.70861576320682107), (1, 0.1431958007198823)]
[(0, -0.42764142348481798), (1, -0.88527674470703799)]
[(0, -0.66124862582594512), (1, 0.4190711252114323)]
```

因此第一三两句和主题一相似，第二句和主题二相似

同理做个LDA

```
lda = models.LdaModel(corpus_tfidf, id2word=dic, num_topics=2)
```

```
ldaOut=lda.print_topics(2)
```

```
print ldaOut[0]
```

```
print ldaOut[1]
```

```
corpus_lda = lda[corpus_tfidf]
```

```
for doc in corpus_lda:
```

```
    print doc
```

得到的结果每次都变，给一次的输出：

```
0.077*吃 + 0.075*北京 + 0.075*雾 + 0.074*今天 + 0.073*不 + 0.072*霾 + 0.070*喜欢 + 0.068*我 + 0.062*的 + 0.061*土豆
0.091*吃 + 0.073*搭 + 0.073*土豆 + 0.073*个 + 0.073*是 + 0.072*百 + 0.071*东西 + 0.066*我 + 0.065*喜欢 + 0.059*霾
```

```
[(0, 0.31271095988105352), (1, 0.68728904011894654)]
```

```
[(0, 0.19957991735916861), (1, 0.80042008264083142)]
```

```
[(0, 0.80940337254233863), (1, 0.19059662745766134)]
```

第一二句和主题二相似，第三句和主题一相似



4



结论和LSI不一样，我估计这和样本数目太少，区别度不高有关，毕竟让我来区分把第一句和哪一句分在一个主题，我也不确定

输入一句话，查询属于LSI得到的哪个主题类型，先建立索引：

```
index = similarities.MatrixSimilarity(lsi[corpus])
query = "雾霾"
query_bow = dic.doc2bow(list(jieba.cut(query)))
print query_bow
query_lsi = lsi[query_bow]
print query_lsi
输出:
[(13, 1), (14, 1)]
[(0, 0.50670602027401368), (1, -0.3678056037187441)]
与第一个主题相似
```

比较和第几句话相似，用LSI得到的索引接着做，并排序输出

```
sims = index[query_lsi]
print list(enumerate(sims))
sort_sims = sorted(enumerate(sims), key=lambda item: -item[1])
print sort_sims
输出：
```

```
[(0, 0.90161765), (1, -0.10271341), (2, 0.99058259)]
```

```
[(2, 0.99058259), (0, 0.90161765), (1, -0.10271341)]
```

可见和第二句话相似度很高，因为只有第二句话出现了雾霾两个词，可是惊讶的是和第一句话的相似度也很高，这得益于LSI模型的算法：**在A和C共现，B和C共现的同时，可以找到A和B的相似度**

在此感谢52nlp的好资源



发表你的评论

([http://my.csdn.net/weixin\\_35068028](http://my.csdn.net/weixin_35068028))



u014386870 (/u014386870) 2017-07-04 13:45

4楼

(/u014386870)

[html]

01. <h1>专门登录顶一个</h1>

回复



u013700085 (/u013700085) 2016-01-05 11:40

3楼

(/u013700085) 在

将文章投影到主题空间中

```
corpus_lsi = lsi[corpus_tfidf]
for doc in corpus_lsi:
    . . .
```



4



print doc

输出：

```
[(0, -0.70861576320682107), (1, 0.1431958007198823)]
[(0, -0.42764142348481798), (1, -0.88527674470703799)]
[(0, -0.66124862582594512), (1, 0.4190711252114323)]
```

这一步中 可能会出现 某个文档和某个主题之间的投影关系确实 比如 上面的矩阵可能会变成：

```
[(0, -0.70861576320682107), (1, 0.1431958007198823)]
[(0, -0.42764142348481798), (1, -0.88527674470703799)]
[(0, -0.66124862582594512)]
```

或者：

```
[(0, -0.70861576320682107), (1, 0.1431958007198823)]
[(0, -0.42764142348481798), (1, -0.88527674470703799)]
[(1, 0.4190711252114323)]
```

是不是说明 对应文档和在对应主题上不存在投影，也就是说相似度其实是-1 呢

回复

twenty\_first (/twenty\_first)

2014-07-16 13:18

2楼

(/twenty\_first)similarities.SparseMatrixSimilarity(tfidf[corpus], num\_features=14), num\_features 这个值最好和字典的大小一样。 num\_features is the number of features in the corpus (e.g. size of the dictionary, or the number of latent topics for latent semantic models).

回复

1条回复

查看 8 条热评

相关文章推荐

gensim文本主题模型推荐 (http://blog.csdn.net/fenxishichengzhang/article/details/517469...

用gensim包做中文文本的推荐 一、gensim是generate similar的简写，叫做普遍相似。对于gensim这个包建议新手直接使用anaconda工具进行集中安装 二、gensim包...

fenxishichengzhang (http://blog.csdn.net/fenxishichengzhang) 2016年06月23日 21:51 571

nlTK在python中的安装,以及nlTK的data库 (http://blog.csdn.net/elikai/article/details/46848671)

最近开始学习Python+NLTK自然语言处理，在此分享自己的学习经验，因为是初学，肯定有很多很多不懂的地方，发布此文章绝非为了显示自己的水平，而是因为网络上对NLTK的资料实在太少了，我就想分享一下...

elikai (http://blog.csdn.net/elikai) 2015年07月12日 10:37 5904

霸气！重磅改革！吴恩达说：女儿识字后就教她学Python！

Python的火爆最近越来越挡不住了，连身边多年工作经验的朋友都开始学Python了！他是这么说的....

(http://www.baidu.com/cb.php?c=lgF\_pyfqHmknjnvPjc0IZ0qnfk9ujYzP1ndPWb10Aw-5Hc3rHnYnHb0TAq15HfLPWRznjb0T1dhuhFbrjDdn1bdrHnkPWfL0AwY5HDdnHnYnjT4PHb0lgF\_5y9YIZ0lQzq-uZR8mLPbUB48ugfElAqspynETZ-YpAq8nWqdlAdxTvqdThP-5yF\_UvTKn0KzujYk0AFV5H00TZcqN0KdpYfqHRLPjnvnfKEpyfqHc4rj6kP0KWpyfqP1cwrHnz0AqLUWYs0ZK45HcsP6KWThnqPWn3r0)

4

http://blog.csdn.net/whzhcahzxh/article/details/17528261

6/9



**基于gensim的文本主题模型(LDA)分析** (<http://download.csdn.net/download...>)

<http://download.csdn.net/download...> 2016年01月05日 20:52 19.53MB 

下载

**主题模型TopicModel：通过gensim实现LDA** (<http://blog.csdn.net/pipisorry/article/details/4...>)


<http://blog.csdn.net/pipisorry/article/details/46447561>使用python gensim轻松实现lda模型。gensim简介Gensim是一个相当专...

 pipisorry (<http://blog.csdn.net/pipisorry>) 2015年06月10日 22:27 

8190

**GENSIM 使用笔记2 — 主题模型和相似性查询** (<http://blog.csdn.net/MebiuW/article/details/5...>)

GENSIM 使用笔记1 — 语料和向量空间 GENSIM 使用笔记2 — 主题模型和相似性查询 在上一个笔记当中，使用gensim针对中文预料创建了字典和语料库，在这一章节中，主要讲下如何创建相...

 MebiuW (<http://blog.csdn.net/MebiuW>) 2016年12月25日 17:04 

1976



**区块链与程序员：赚钱还是创业**

比特币和区块链是信息技术发展的产物，只有程序猿能够深入到代码层。与其他人相比，程序员们对区块链理解更深刻，投资更可能成功。你将获得巨大的认知快感：还能这么玩啊，为啥我没有想到！

([http://www.baidu.com/cb.php?c=lgF\\_pyfqhHmknjTYPHR0IZ0qnfK9ujYzP1f4Pjn10Aw-5Hc4nj6vPjm0TAq15Hf4rjn1n1b0T1YkPhcvnhf4mhwrHmvPyFb0AwY5HDdnHnYnjT4PW00lgF\\_5y9YIZ0IQzqMpgwBUvqoQhP8QvIGIAPCmgfEmvq\\_ljd8Q1N9nyuWrAcYnWDkuhDYN1Nbnhc4uHbYm1qdlAdxTvqdThP-5HDznHNWmhkEusKzujYk0AFV5H00TZcqn0KdpyfqhHRLPjnvnfKEpyfqhHnsnj0YnsKWpyfqP1cvrHnz0AqLUWYs0ZK45HcsP6KWThnqPWcLns](http://www.baidu.com/cb.php?c=lgF_pyfqhHmknjTYPHR0IZ0qnfK9ujYzP1f4Pjn10Aw-5Hc4nj6vPjm0TAq15Hf4rjn1n1b0T1YkPhcvnhf4mhwrHmvPyFb0AwY5HDdnHnYnjT4PW00lgF_5y9YIZ0IQzqMpgwBUvqoQhP8QvIGIAPCmgfEmvq_ljd8Q1N9nyuWrAcYnWDkuhDYN1Nbnhc4uHbYm1qdlAdxTvqdThP-5HDznHNWmhkEusKzujYk0AFV5H00TZcqn0KdpyfqhHRLPjnvnfKEpyfqhHnsnj0YnsKWpyfqP1cvrHnz0AqLUWYs0ZK45HcsP6KWThnqPWcLns))

**初试主题模型LDA-基于python的gensim包** ([http://blog.csdn.net/a\\_step\\_further/article/deta...](http://blog.csdn.net/a_step_further/article/deta...))


LDA是文本挖掘中常用的主题模型，用来从大量文档中提取出最能表达各个主题的一些关键词，具体算法原理可参阅KM上相关文章。笔者因业务需求，需对腾讯微博上若干账号的消息进行主题提取，故而尝试了一下该算法，...

 a\_step\_further ([http://blog.csdn.net/a\\_step\\_further](http://blog.csdn.net/a_step_further)) 2016年04月18日 08:11 

6605

**主题模型Python工具包：Gensim** (<http://blog.csdn.net/aihali/article/details/45044299>)

Gensim是一个相当专业的主题模型Python工具包。在文本处理中，比如商品评论挖掘，有时需要了解每个评论分别和商品的描述之间的相似度，以此衡量评论的客观性。评论和商品描述的相似度越高，说明评论的用...

 aihali (<http://blog.csdn.net/aihali>) 2015年04月14日 16:46 

725

**gensim 主题模型 seed** ([http://blog.csdn.net/Tiffany\\_Li2015/article/details/49249957](http://blog.csdn.net/Tiffany_Li2015/article/details/49249957))


1、关于gensim LDA主题模型网上有个简单的的例子：<http://blog.itpub.net/16582684/viewspace-1253901/>我也用来在自己集子上试验了下效果。 我的数...

 Tiffany\_Li2015 ([http://blog.csdn.net/Tiffany\\_Li2015](http://blog.csdn.net/Tiffany_Li2015)) 2015年10月19日 14:41 

513

**文本分析--基于gensim的文本主题模型分析** (<http://blog.csdn.net/kevinelstri/article/details/7...>)

```
#!/usr/bin/python # -*- coding:utf8 -*-import os import time import re import jieba.analyse import t...
```

 kevinelstri (<http://blog.csdn.net/kevinelstri>) 2017年04月12日 18:05 0677


**基于gensim的文本主题模型(LDA)分析** (<http://blog.csdn.net/u010297828/article/details/504...>)

主题模型文本分析小例子

 u010297828 (<http://blog.csdn.net/u010297828>) 2016年01月05日 20:56 7066

**利用gensim主题模型寻找相似的coursera课程** (<http://blog.csdn.net/kesonyk/article/details/...>)

参考[#encoding=utf-8 from nltk.tokenize import word\\_tokenize from nl...](http://www.52nlp.cn/如何计算两个文档的相似度三)

 kesonyk (<http://blog.csdn.net/kesonyk>) 2015年06月26日 15:08 430

**用 LDA 做主题模型：当 MLlib 邂逅 GraphX** (<http://blog.csdn.net/bbbeoy/article/details/715...>)

主题模型可以从一系列文章中自动推测讨论的主题。这些主题可以被用作总结和整理文章，也可以在机器学习流程的后期阶段用于特征化和降维。在Spark 1.3中，MLlib现在支持最成功的主题模型之一，...

 bbbeoy (<http://blog.csdn.net/bbbeoy>) 2017年05月10日 10:44 122

**用 LDA 做主题模型：当 MLlib 邂逅 GraphX** ([http://blog.csdn.net/qq\\_27231343/article/detai...](http://blog.csdn.net/qq_27231343/article/detai...))

主题模型可以从一系列文章中自动推测讨论的主题。这些主题可以被用作总结和整理文章，也可以在机器学习流程的后期阶段用于特征化和降维。在Spark 1.3中，MLlib现在支持最成功的主题模型之一，...

 qq\_27231343 ([http://blog.csdn.net/qq\\_27231343](http://blog.csdn.net/qq_27231343)) 2016年07月05日 15:45 244

**用 LDA 做主题模型：当 MLlib 邂逅 GraphX** (<http://blog.csdn.net/huludan/article/details/51...>)


转载： <http://blog.jobbole.com/86130/> 主题模型可以从一系列文章中自动推测讨论的主题。这些主题可以被用作总结和整理文章，也可以在机器学习流程的后期阶段用...

 huludan (<http://blog.csdn.net/huludan>) 2016年07月07日 22:55 552

 **主题模型&code;** ([http://download.csdn.net/download/qq\\_30058597/...](http://download.csdn.net/download/qq_30058597/...))

 2017年12月12日 16:03 10.36MB [下载](#)

 **靳志辉：大规模主题模型建模及其在腾讯业务中的应用** ([http://download.csdn.net/download/qq\\_30058597/...](http://download.csdn.net/download/qq_30058597/...))

 2014年12月18日 14:01 1.86MB [下载](#)

**NLP | LDA主题模型的应用难题、使用心得及从多元统计角度剖析** (<http://blog.csdn.net/sinat...>)

将LDA跟多元统计分析结合起来看，那么LDA中的主题就像词主成分，其把主成分-样本之间的关系说清楚了。多元学的时候

多元统计分析结合来看，那么LDA中的主题就像词主成分，其把主成分-样本之间的关系说清楚了。多元学的时候



- 4
- 
- 
- 

聚类方法K聚类、K均值聚类以及主成分分析。K均值聚类、主成分分析针对变量，K聚类聚类针对特征...

 [sinat\\_26917383](http://blog.csdn.net/sinat_26917383) (http://blog.csdn.net/sinat\_26917383) 2016年08月17日 18:55  4356

---



**主题模型讲义** (http://download.csdn.net/download/xizi\_fish/7629085)

[http://download.csdn.net/download/xizi\\_fish/7629085](http://download.csdn.net/download/xizi_fish/7629085) 2014年07月13日 15:32 31.01MB 

下载

---



**LDA主题模型代码 分词代码** (http://download.csdn.net/download/qq\_2727...) (http://download.csdn.net/download/qq\_2727...)

[http://download.csdn.net/download/qq\\_2727...](http://download.csdn.net/download/qq_2727...) 2015年11月22日 21:44 14.17MB 

下载

---

 [u010533386](http://blog.csdn.net/u010533386) (http://blog.csdn.net/u010533386) 2016年05月18日 05:28  16131

**中文文本预处理--主题模型** (http://blog.csdn.net/u010533386/article/details/51440854)

去掉低频词、分词、繁简转化、替换奇异词等是中文文本数据处理中的重要步骤。...