文本挖掘从小白到精通（五）---主题模型的主题数确定和可视化

AINLP 昨天

以下文章来源于Social Listening与文本挖掘 ， 作者Scottish Fold Cats

Social Listening与文本挖掘

爱好文本分析，社会化媒体数据挖掘

写在前面：笔者最近在梳理自己的文本挖掘知识结构，借助gensim、sklearn、keras等库的文档做了些扩充，会陆陆续续介绍文本向量化、tfidf、主题模型、word2vec，既会涉及理论，也会有详细的代码和案例进行讲解，希望在梳理自身知识体系的同时也能对想学习文本挖掘的朋友有一点帮助，这是笔者写该系列的初衷。

前面几篇文章从词向量空间模型、词袋表示、TF-IDF聊到各类主题模型（LSA、LDA、RP和HDP），再到基于LSA/LSI的文本检索，本文将回到主题模型这个话题中来，聊聊主题模型中的主题数该如何确定，以及主题模型的可视化，请大家enjoy~

温馨提示：图片显示毛糙和不清楚，是分辨率过高的缘故，点击图片，即可看到高清大图。

首先，导入必要的库：

```
from gensim.corpora import Dictionary
from gensim.models import ldamodel
from gensim.models import CoherenceModel, LdaModel
from gensim import models
import numpy
%matplotlib inline
```

在这里，笔者想展示gensim的主题模型中的2个新的方法 --- get_term_topics和get_document_topics，接下来，大家将会看到，在不同的语境中，同一个词汇的意义会不一样的情形（The same word which might have different meanings in different context）。

笔者想以“苹果”一词为例，苹果最常见的含义是水果；另一个含义是苹果公司，该公司拥有世界知名的iPhone和Mac。 在下面的示例数据集中，有13个文档，每个文档经过分词处理和去停用词处理。

根据上述想法，笔者构建了如下语料库，已经经过分词和去停用词处理，短小精悍，用作demo数据刚刚好。

```
texts = [
            ['苹果','叶子','椭圆形','树上'],
            ['植物','叶子','绿色','落叶乔木'],
            ['水果','苹果','红彤彤','味道'],
            ['苹果','落叶乔木','树上','水果'],
            ['植物','营养','水果','维生素'],
            ['营养','维生素','苹果','成分'],
            ['互联网','电脑','智能手机','高科技'],
            ['苹果','公司','互联网','品质'],
            ['乔布斯','苹果','硅谷'],
            ['电脑','智能手机','苹果','乔布斯'],
            ['苹果','电脑','品质','生意'],
            ['电脑','品质','乔布斯'],
            ['苹果','公司','生意','硅谷']

            ]

dictionary = Dictionary(texts)
corpus = [dictionary.doc2bow(text) for text in texts]
```

接下来，笔者将训练两个主题模型，差异在于主题数的不同，按照笔者构建的语料库构成来看，主题数应该是2，假如是其他的主题数，模型的效果应该不好。

下面，基于假设，“好”的主题模型的主题数为2，“坏”的主题模型的主题数为6。

```
numpy.random.seed(1) # 设置随即种子数，以便相同的设置能跑出相同的结果，可复现
goodLdaModel = LdaModel(corpus=corpus, id2word=dictionary,
     iterations=50, num_topics=2)
badLdaModel = LdaModel(corpus=corpus, id2word=dictionary,
     iterations=50, num_topics=6)
```

我们通过CoherenceModel这个类中的两个指标 --- U_Mass Coherence和C_V coherence来判定主题模型质量的好坏（对文本的主题区分度效果，即能将混沌的语料切分出人类可理解的主题），这两个指标都是数值越大，主题模型的效果越好。

1 如何通过指标确定合理的主题数

1.1 使用U_Mass Coherence

```
goodcm = CoherenceModel(model=goodLdaModel, corpus=corpus,
       dictionary=dictionary, coherence='u_mass')
badcm = CoherenceModel(model=badLdaModel, corpus=corpus,
       dictionary=dictionary, coherence='u_mass')
```

```
print(goodcm.get_coherence())
print(badcm.get_coherence())
```

-18.78459503442916
-18.83035774373808

虽然数值差异不大，但仍能看出“好”的主题模型的U_Mass Coherence要大于坏的模型的数值。

1.2 使用 C_V coherence

```
goodcm = CoherenceModel(model=goodLdaModel, texts=texts,
    dictionary=dictionary,  coherence='c_v')
badcm = CoherenceModel(model=badLdaModel, texts=texts,
    dictionary=dictionary,  coherence='c_v')
```

```
print(goodcm.get_coherence())
print(badcm.get_coherence())
```

0.5880602397643417
0.5868870960312388

跟上述结果一样，虽然数值差异不大，但“好”的主题模型的C_V coherence值仍要大于“坏”的主题模型的Coherence值。

由此，我们即可知道主题数应该设置为2，以此期望主题模型能体现“苹果”一词的多义性---一个跟（苹果）公司有关，一个跟水果有关。

```
model = ldamodel.LdaModel(corpus, id2word=dictionary,
            iterations=500,num_topics=2,alpha='auto')
```

展示两个主题中的主题词，由之推断出主题的大致内容：

```
model.show_topics(num_words=5)
```

[(0, '0.143"苹果" + 0.077*"水果" + 0.070*"树上" + 0.058*"电脑" + 0.048*"叶子"'),*
(1, '0.135"苹果" + 0.078*"电脑" + 0.075*"乔布斯" + 0.074*"品质" + 0.055*"植物"')]*

正如我们所期望的那样，LDA模型给了我们可接受的结果（限于语料库太过稀少，主题模型并未发挥真正的功力！）。 正如我们所看到的，“苹果”是这两个主题中最有影响力的词汇（主题权重值在两个主题中都是最大，分别为0.143和0.135）。

两个主题中除“苹果”以外的词汇相当于“语境”，有助于我们分辨各个主题下的“苹果”的具体含义，是指水果还是指乔帮主所创立的公司。

接下来，对主题模型跑出来的结果进行深入分析，试试 get_term_topics和get_document_topics这两个新方法。

2 预测词汇的主题归属

函数get_term_topics返回属于某个词汇属于特定主题的几率（the odds of that particular word belonging to a particular topic）。
举3个个例子：

```
'''
[(0,
  '0.104*"苹果" + 0.077*"水果" + 0.073*"植物" + 0.071*"落叶乔木" + 0.070*"叶子" +
 (1,
  '0.175*"苹果" + 0.084*"电脑" + 0.081*"品质" + 0.062*"乔布斯" + 0.061*"生意" + 0
 '''

#根据主题模型运行出来的结果，序号为0的暂定为“水果”，序号为1的暂定为“公司”，用来测试几个词
topic_list = ['水果','公司']
```

```
[(topic_list[i[0]],i[1]) for i in model.get_term_topics('树上')]
```

[('水果', 0.05612464)]

显而易见，“树上”这个词汇更加靠近“水果”这个主题。

```
[(topic_list[i[0]],i[1]) for i in model.get_term_topics('智能手机')]
```

[('水果', 0.03219079), ('公司', 0.056897775)]

“智能手机”当然离“公司”这个话题更近一些~

```
[(topic_list[i[0]],i[1]) for i in model.get_term_topics('落叶乔木')]
```

[('水果', 0.039815596), ('公司', 0.028762106)]

“落叶乔木”这个词汇的更“亲近”于“水果”这个话题。

3 预测文档的主题归属

get_document_topics是一个用于推断文档主题归属的函数/方法，在这里，假设一个文档可能同时包含若干个主题，但在每个主题上的概率不一样，文档最有可能从属于概率最大的主题。

此外，该函数也可以让我们了解某个文档中的某个词汇在主题上的分布情况。

现在让我们来测试下，两个包含“苹果”的语句的主题从属情况，这两个语句已经经过分词和去停用词处理，仅反映语句主干信息，每个语句中除“苹果”以外的词汇可以看作是它的“（上下文）语境”，由语境我们可以推断出“苹果”究竟是哪种含义。

当参数per_word_topics设置为True时，get_document_topics方法返回词汇ID（在词典中的位置），以及最可能的主题id的列表（按降序排列，可能性最大的排在前面）。

```
bow_fruit = ['苹果','树上','落叶乔木','苹果']
bow_company = ['苹果','电脑','乔布斯']
```

```
bow = model.id2word.doc2bow(bow_fruit)      # 现将文档转换为词袋表示
doc_topics, word_topics, phi_values = model.get_document_topics(bow, per_word_t

word_topics
```

[(1, [0, 1]), (3, [0, 1]), (6, [0, 1])]

上面的结果该如何解读呢？它应该这样理解：1、3、6对应文档bow_fruit中的3个词汇：'苹果'、'树上'、'落叶乔木'，它们的主题更加倾向于是“水果”，因为每个词汇的主题序号中，0都排在靠前的位置，也就是“水果”这个主题更为明显。

大家可能注意到了，get_document_topics这个方法产生了3个值doc_topics、 word_topics和phi_values。 对于phi_values而言，它包含特定词汇在各主题上的phi值，且按特征长度缩放。 Phi本质上是文档中某个词汇从属于某个主题的概率值。笔者将会在下面的例子中说明这一点。

```
phi_values
```

```
[(1, [(0, 0.9572513), (1, 0.042745788)]),
 (3, [(0, 1.5801868), (1, 0.4198111)]),
 (6, [(0, 0.78943187), (1, 0.21056366)])]
```

上述结果其实跟word_topics的结果的结果差不多，除了各个主体序号后多出的数值---phi_values，有了这个数值，我们能判断词汇从属于特定主题的程度如何。 值得注意的词汇3,也就是“苹果”这个词汇，因为它在文档中出现了2次，我们可以看到按特征长度的缩放非常明显。 phi_values的总和是2，而不是1。

现在我们确切地知道了get_document_topics是用来干嘛的，现在让我们对第二个文档bow_company做同样的事情。

```
bow = model.id2word.doc2bow(bow_company)  # 现将文档转换为词袋表示
doc_topics, word_topics, phi_values = model.get_document_topics(bow, per_word_t

word_topics
```

```
[(3, [1, 0]), (15, [1, 0]), (19, [1, 0])]
```

因为“苹果”这个词现在用于“公司”这个背景之下，此时它更可能与“topic_1”有关。

我们已经非常清楚地看到，基于不同的上下文，同一个词汇会归属到不同的主题下。 这与我们之前的get_term_topics方法不同，它是一个'**静态（Static）**'的**主题分布（Topic Distribution）**。

我们还必须注意这一点 --- 因为基于gensim实现的LDA使用**变分贝叶斯采样（Variational Bayes Sampling）**，所以某个文档中出现多次的某个词汇的仅给出一个主题分布。例如，对于句子“苹果长在树上，秋天上面会结满红彤彤的苹果”，其中的两个“苹果”都会被分配到主题“topic_0（水果）”上，也就是说，这两个“苹果”具有相同的主题分布。

接下来，笔者通过get_document_topics来获取语料库中所有文档的“doc_topics”，“word_topics”和“phi_values”：

```
all_topics = model.get_document_topics(corpus, per_word_topics=True)

cnt = 0
for doc_topics, word_topics, phi_values in all_topics:
```

```
    print('新文档:{} \n'.format(cnt),texts[cnt])
    doc_topics = [(topic_list[i[0]],i[1]) for i in doc_topics]
    word_topics = [(dictionary.id2token [i[0]],i[1]) for i in word_topics]
    phi_values = [(dictionary.id2token [i[0]],i[1]) for i in phi_values ]
    print('文档主题:', doc_topics)
    print('词汇主题:', word_topics)
    print('Phi值:', phi_values)
    print(" ")
    print('-------------- \n')
    cnt+=1
```

新文档:0
['苹果', '叶子', '椭圆形', '树上']
文档主题: [('水果', 0.80287796), ('公司', 0.19712202)]
词汇主题: [('叶子', [0, 1]), ('树上', [0, 1]), ('椭圆形', [0, 1]), ('苹果', [0, 1])]
Phi值: [('叶子', [(0, 0.8883188), (1, 0.11167712)]), ('树上', [(0, 0.97451615), (1, 0.02548

新文档:1
['植物', '叶子', '绿色', '落叶乔木']
文档主题: [('水果', 0.19830076), ('公司', 0.8016992)]
词汇主题: [('叶子', [1, 0]), ('植物', [1, 0]), ('绿色', [1, 0]), ('落叶乔木', [1, 0])]
Phi值: [('叶子', [(0, 0.17412838), (1, 0.82586676)]), ('植物', [(0, 0.040288094), (1, 0.959

新文档:2
['水果', '苹果', '红彤彤', '味道']
文档主题: [('水果', 0.8195776), ('公司', 0.18042246)]
词汇主题: [('苹果', [0, 1]), ('味道', [0, 1]), ('水果', [0, 1]), ('红彤彤', [0, 1])]
Phi值: [('苹果', [(0, 0.8836959), (1, 0.1163031)]), ('味道', [(0, 0.97444284), (1, 0.025552

新文档:3
['苹果', '落叶乔木', '树上', '水果']
文档主题: [('水果', 0.78223956), ('公司', 0.21776046)]
词汇主题: [('树上', [0, 1]), ('苹果', [0, 1]), ('落叶乔木', [0, 1]), ('水果', [0, 1])]
Phi值: [('树上', [(0, 0.9694993), (1, 0.030498091)]), ('苹果', [(0, 0.84234864), (1, 0.1576

新文档:4

['植物', '营养', '水果', '维生素']

文档主题:[('水果', 0.25157169), ('公司', 0.7484283)]
词汇主题:[('植物', [1, 0]), ('水果', [1, 0]), ('维生素', [1, 0]), ('营养', [1, 0])]
Phi值:[('植物', [(0, 0.06110432), (1, 0.9388922)]), ('水果', [(0, 0.3684698), (1, 0.631526

新文档:5
['营养', '维生素', '苹果', '成分']
文档主题:[('水果', 0.24819611), ('公司', 0.7518039)]
词汇主题:[('苹果', [1, 0]), ('维生素', [1, 0]), ('营养', [1, 0]), ('成分', [1, 0])]
Phi值:[('苹果', [(0, 0.20490424), (1, 0.7950948)]), ('维生素', [(0, 0.08610094), (1, 0.913

新文档:6
['互联网', '电脑', '智能手机', '高科技']
文档主题:[('水果', 0.6843853), ('公司', 0.3156147)]
词汇主题:[('互联网', [0, 1]), ('智能手机', [0, 1]), ('电脑', [0, 1]), ('高科技', [0, 1])]
Phi值:[('互联网', [(0, 0.7354029), (1, 0.26459274)]), ('智能手机', [(0, 0.7563021), (1, 0.

新文档:7
['苹果', '公司', '互联网', '品质']
文档主题:[('水果', 0.20751746), ('公司', 0.7924826)]
词汇主题:[('苹果', [1, 0]), ('互联网', [1, 0]), ('公司', [1, 0]), ('品质', [1, 0])]
Phi值:[('苹果', [(0, 0.156428), (1, 0.843571)]), ('互联网', [(0, 0.16034536), (1, 0.839650

新文档:8
['乔布斯', '苹果', '硅谷']
文档主题:[('水果', 0.20053746), ('公司', 0.7994625)]
词汇主题:[('苹果', [1, 0]), ('乔布斯', [1, 0]), ('硅谷', [1, 0])]
Phi值:[('苹果', [(0, 0.13529739), (1, 0.86470157)]), ('乔布斯', [(0, 0.038489204), (1, 0.9

新文档:9
['电脑', '智能手机', '苹果', '乔布斯']
文档主题:[('水果', 0.22308852), ('公司', 0.7769115)]
词汇主题:[('苹果', [1, 0]), ('智能手机', [1, 0]), ('电脑', [1, 0]), ('乔布斯', [1, 0])]
Phi值:[('苹果', [(0, 0.17488222), (1, 0.82511675)]), ('智能手机', [(0, 0.19596255), (1, 0.

新文档:10

```
['苹果', '电脑', '品质', '生意']
文档主题:[('水果', 0.17786688), ('公司', 0.8221331)]
词汇主题:[('苹果', [1, 0]), ('电脑', [1, 0]), ('品质', [1, 0]), ('生意', [1, 0])]
Phi值:[('苹果', [(0, 0.121820726), (1, 0.8781783)]), ('电脑', [(0, 0.080861986), (1, 0.919

----------

新文档:11
['电脑', '品质', '乔布斯']
文档主题:[('水果', 0.18384826), ('公司', 0.81615174)]
词汇主题:[('电脑', [1, 0]), ('品质', [1, 0]), ('乔布斯', [1, 0])]
Phi值:[('电脑', [(0, 0.07663635), (1, 0.92336154)]), ('品质', [(0, 0.035397064), (1, 0.964

----------

新文档:12
['苹果', '公司', '生意', '硅谷']
文档主题:[('水果', 0.17403097), ('公司', 0.82596904)]
词汇主题:[('苹果', [1, 0]), ('公司', [1, 0]), ('硅谷', [1, 0]), ('生意', [1, 0])]
Phi值:[('苹果', [(0, 0.11741197), (1, 0.882587)]), ('公司', [(0, 0.06634161), (1, 0.933654

----------
```

如果你想在变量中存储语料库中所有文档的“doc_topics”、“word_topics”和“phi_values”，以及后续使用这3个数据来获取特定文档的详细信息，可以按以下方式实现：

```
topics = model.get_document_topics(corpus, per_word_topics=True)
all_topics = [(doc_topics, word_topics, word_phis) for doc_topics, word_topics,
```

现在，我可以访问特定文档的详细信息，如文档['苹果', '落叶乔木', '树上', '水果']，结果如下所示：

```
doc_topic, word_topics, phi_values = all_topics[2]
print('文档主题:', doc_topics, "\n")
print('词汇主题:', word_topics, "\n")
print('Phi值:', phi_values)
```

```
文档主题:[('水果', 0.17403097), ('公司', 0.82596904)]

词汇主题:[(3, [0, 1]), (7, [0, 1]), (8, [0, 1]), (9, [0, 1])]

Phi值:[(3, [(0, 0.8837178), (1, 0.116281204)]), (7, [(0, 0.97444814), (1, 0.02554696)]), (
```

我们也可以通过以下方式打印所有文档的详细信息：

```
cnt = 0
for doc in all_topics:
  print('新文档:{} \n'.format(cnt),texts[cnt])
  print('文档主题:', doc[0])
  print('词汇主题:', doc[1])
  print('Phi值:', doc[2])
  print(" ")
  print('-------------- \n')
  cnt+=1
```

新文档:0
['苹果', '叶子', '椭圆形', '树上']
文档主题:[(0, 0.8028161), (1, 0.19718389)]
词汇主题:[(0, [0, 1]), (1, [0, 1]), (2, [0, 1]), (3, [0, 1])]
Phi值:[(0, [(0, 0.88826054), (1, 0.11173541)]), (1, [(0, 0.97450155), (1, 0.025495822)]),

----

新文档:1
['植物', '叶子', '绿色', '落叶乔木']
文档主题:[(0, 0.19823843), (1, 0.80176157)]
词汇主题:[(0, [1, 0]), (4, [1, 0]), (5, [1, 0]), (6, [1, 0])]
Phi值:[(0, [(0, 0.1740438), (1, 0.8259514)]), (4, [(0, 0.040265355), (1, 0.9597313)]), (5,

----

新文档:2
['水果', '苹果', '红彤彤', '味道']
文档主题:[(0, 0.81959766), (1, 0.18040234)]
词汇主题:[(3, [0, 1]), (7, [0, 1]), (8, [0, 1]), (9, [0, 1])]
Phi值:[(3, [(0, 0.8837178), (1, 0.116281204)]), (7, [(0, 0.97444814), (1, 0.02554696)]), (

新文档:3
['苹果', '落叶乔木', '树上', '水果']
文档主题:[(0, 0.78215605), (1, 0.21784389)]
词汇主题:[(1, [0, 1]), (3, [0, 1]), (6, [0, 1]), (8, [0, 1])]
Phi值:[(1, [(0, 0.9694783), (1, 0.030518971)]), (3, [(0, 0.8422549), (1, 0.15774421)]), (6

新文档:4
['植物', '营养', '水果', '维生素']
文档主题:[(0, 0.25165194), (1, 0.748348)]
词汇主题:[(4, [1, 0]), (8, [1, 0]), (10, [1, 0]), (11, [1, 0])]
Phi值:[(4, [(0, 0.06113779), (1, 0.93885875)]), (8, [(0, 0.3686055), (1, 0.63139063)]), (1

新文档:5
['营养', '维生素', '苹果', '成分']
文档主题:[(0, 0.24821556), (1, 0.75178444)]
词汇主题:[(3, [1, 0]), (10, [1, 0]), (11, [1, 0]), (12, [1, 0])]
Phi值:[(3, [(0, 0.20492749), (1, 0.79507136)]), (10, [(0, 0.086112194), (1, 0.9138841)]),

---

新文档:6
['互联网', '电脑', '智能手机', '高科技']
文档主题:[(0, 0.6843576), (1, 0.31564245)]
词汇主题:[(13, [0, 1]), (14, [0, 1]), (15, [0, 1]), (16, [0, 1])]
Phi值:[(13, [(0, 0.7353709), (1, 0.26462474)]), (14, [(0, 0.7562719), (1, 0.24372388)]), (

新文档:7
['苹果', '公司', '互联网', '品质']
文档主题:[(0, 0.20750786), (1, 0.79249215)]
词汇主题:[(3, [1, 0]), (13, [1, 0]), (17, [1, 0]), (18, [1, 0])]
Phi值:[(3, [(0, 0.15641662), (1, 0.8435824)]), (13, [(0, 0.16033378), (1, 0.83966166)]), (

新文档:8
['乔布斯', '苹果', '硅谷']
文档主题:[(0, 0.20051865), (1, 0.7994814)]
词汇主题:[(3, [1, 0]), (19, [1, 0]), (20, [1, 0])]
Phi值:[(3, [(0, 0.1352751), (1, 0.8647239)]), (19, [(0, 0.038482144), (1, 0.96151555)]), (

新文档:9
['电脑', '智能手机', '苹果', '乔布斯']
文档主题:[(0, 0.22314881), (1, 0.7768512)]
词汇主题:[(3, [1, 0]), (14, [1, 0]), (15, [1, 0]), (19, [1, 0])]

Phi值:[(3, [(0, 0.1749539), (1, 0.825045)]), (14, [(0, 0.19604084), (1, 0.8039544)]), (15,

---

```
新文档:10
['苹果', '电脑', '品质', '生意']
文档主题:[(0, 0.1778042), (1, 0.82219577)]
词汇主题:[(3, [1, 0]), (15, [1, 0]), (18, [1, 0]), (21, [1, 0])]
Phi值:[(3, [(0, 0.121748514), (1, 0.8782505)]), (15, [(0, 0.08081183), (1, 0.9191862)]), (

----------

新文档:11
['电脑', '品质', '乔布斯']
文档主题:[(0, 0.18385334), (1, 0.8161466)]
词汇主题:[(15, [1, 0]), (18, [1, 0]), (19, [1, 0])]
Phi值:[(15, [(0, 0.07664047), (1, 0.9233575)]), (18, [(0, 0.035399046), (1, 0.96459866)]),

----------

新文档:12
['苹果', '公司', '生意', '硅谷']
文档主题:[(0, 0.17405325), (1, 0.82594675)]
词汇主题:[(3, [1, 0]), (17, [1, 0]), (20, [1, 0]), (21, [1, 0])]
Phi值:[(3, [(0, 0.117437586), (1, 0.8825614)]), (17, [(0, 0.06635692), (1, 0.9336393)]), (

----------
```

4 对特定的主题词“着色”

当我们想要为语料库或文档中的某个词汇着色时，我们可以使用一些小技巧。 如果想给语料库中的某个词汇着色，那么get_term_topics将是最好的选择。 如果没有，用get_document_topics也行。

笔者现在尝试使用matplotlib对某些词汇进行着色。 这只是绘制单词的一种方式 - 有更多更好的方法。 比如，[WordCloud]（https://github.com/amueller/word_cloud） 就是这样一个实现词汇可视化的python包。

为简单起见，笔者将topic_1设定为红色，将topic_0设定为蓝色。

```
#  以下是一个给词汇着色的函数。 如前所述，有很多方法可以做到这一点。
import matplotlib.pyplot as plt
import matplotlib.patches as patches
def color_words(model, doc):
    # 将文档转化为词袋表示
    doc = model.id2word.doc2bow(doc)
    # 获取词汇的主题分布
```

```
    doc_topics, word_topics, phi_values = model.get_document_topics(doc, per_wor

    # 颜色 - 主题匹配
    topic_colors = { 1:'red', 0:'blue'}

    # 设置画图的背景
    fig = plt.figure()
    ax = fig.add_axes([0,0,1,1])

    # 一个“暗黑”小技巧，使词汇之间的间距保持在合理的范围之内
    word_pos = 1/len(doc)

   # 使用matplotlib对词汇进行绘制
   for word, topics in word_topics:
     ax.text(word_pos,
                  0.8,
                  model.id2word[word],
                  horizontalalignment='center',
                  verticalalignment='center',
                  fontsize=20,
                  color=topic_colors[topics[0]],  # 选择可能性最大的主题
                  transform=ax.transAxes)
         word_pos += 0.2
      ax.set_axis_off()
      plt.show()
```

现在开始对一些文档进行可视化展示，试图发现词汇和主题的关联性：

```
# 对“水果”主题的文档进行可视化展示

import matplotlib.pyplot as plt

%matplotlib inline
plt.rcParams['font.sans-serif']=['SimHei']  #用来正常显示中文标签

bow_fruit = ['水果','苹果','树上']

color_words(model, bow_fruit)
```


```
bow_company = ['苹果','电脑','乔布斯','智能手机']
color_words(model, bow_company)
```

这里有趣的是，虽然“苹果”在第一个语句中显示为蓝色，但由于这里是“公司”相关的语境，它现在又显示为红色 --- 之前的主题权重值已经证明了这一点。

```
#  现在找一个主题分布较为均匀的语句，看看这里的可视化效果如何

doc = ['苹果','电脑','苹果','树上','乔布斯','智能手机','硅谷','植物']
color_words(model, doc)
```

这里，我们看到文档的词汇着色基本我们预期的（“植物”没分对）。:)

下面，笔者再对整个词典（dictionary）里的词汇进行着色。

在这里，与上面使用get_document_topics不同的是，我们使用get_term_topics这个方法，对整个字典（dictionary）里的词汇进行颜色区分，蓝色的代表主题0 - “水果”，红色代表主题1 - “公司”。

```
import matplotlib.pyplot as plt
import matplotlib.patches as patches
def color_words_dict(model, dictionary):
  word_topics = []
  for word_id in dictionary:
    word = str(dictionary[word_id])
    # get_term_topics方法返回静态的主题
    probs = model.get_term_topics(word)
       # 我们正在创建词汇的主题分布，它与get_document_topics所结果创建的类似
    try:
       if probs[0][1] >= probs[1][1]:
          word_topics.append((word_id, [0, 1]))
       else:
          word_topics.append((word_id, [1, 0]))
        # 在这种情况下，只会返回一个主题（概率值最大的那个主题）
   except IndexError:
     word_topics.append((word_id, [probs[0][0]]))

    # 颜色 - 主题匹配
    topic_colors = { 1:'red', 0:'blue'}

     # 设置画图的背景
    fig = plt.figure()
    ax = fig.add_axes([0,0,1,1])

    # 一个“暗黑”小技巧，使词汇之间的间距保持在合理的范围之内
    word_pos = 1/len(doc)

    #使用matplotlib对词汇进行绘制
    for word, topics in word_topics:
      ax.text(word_pos, 0.8, model.id2word[word],
            horizontalalignment='center',
            verticalalignment='center',
            fontsize=20,
            color=topic_colors[topics[0]],  # 选择可能性最大的主题
            transform=ax.transAxes)
      word_pos += 0.2

    ax.set_axis_off()
```

```
    plt.show()
```

```
color_words_dict(model, dictionary)
```

正如我们所看到的，大部分红色词汇与“公司”这个主题有关，蓝色词语与“水果”主题有关。

然鹅，您还可以注意到某些单词（如“叶子”、“树上”、“互联网”、“智能手机”等词汇）似乎颜色不正确（也就是主题没划分对）。

由此，笔者不得不提到LDA模型的使用场景及需要注意的问题：

- 主题模型需要大量的文本数据
- 适合长文本，比如文章的正文数据，不适合微博、评论这样的短文本数据
- 小语料库意味着LDA算法可能不会为每个单词分配较为理想的主题比例
- 微调模型的参数并收集更多的语料库将提升模型表现，最终改善词汇着色的结果。

5 主题模型可视化

pyLDAvis是python中的一个对LDA主题模型进行交互可视化的库，它可以将主题模型建模后的结果，制作成一个网页交互版的结果分析工具。

pyLDAvis在可视化呈现中着重回答如下三个问题：

- What is the meaning of each topic? 每个主题的意义是什么?
- How prevalent is each topic? 每个主题在总语料库的比重如何?
- How do the topics relate to each other? 主题之间有什么关联?

以下是实现代码：

```
import warnings
try:
  import pyLDAvis.gensim
  CAN_VISUALIZE = True
  pyLDAvis.enable_notebook()
  from IPython.display import display
except ImportError:
    ValueError("SKIP: please install pyLDAvis")
```

```
    CAN_VISUALIZE = False
warnings.filterwarnings('ignore')  # 忽视所有的提示
%matplotlib inline

prepared = pyLDAvis.gensim.prepare(model, corpus, dictionary)
pyLDAvis.show(prepared,open_browser=True)
```

Note: if you're in the IPython notebook, pyLDAvis.show() is not the best command to use. Consider using pyLDAvis.display(), or pyLDAvis.enable_notebook(). See more information at http://pyLDAvis.github.io/quickstart.html .

You must interrupt the kernel to end this command

Serving to http://127.0.0.1:8889/ [Ctrl-C to exit]

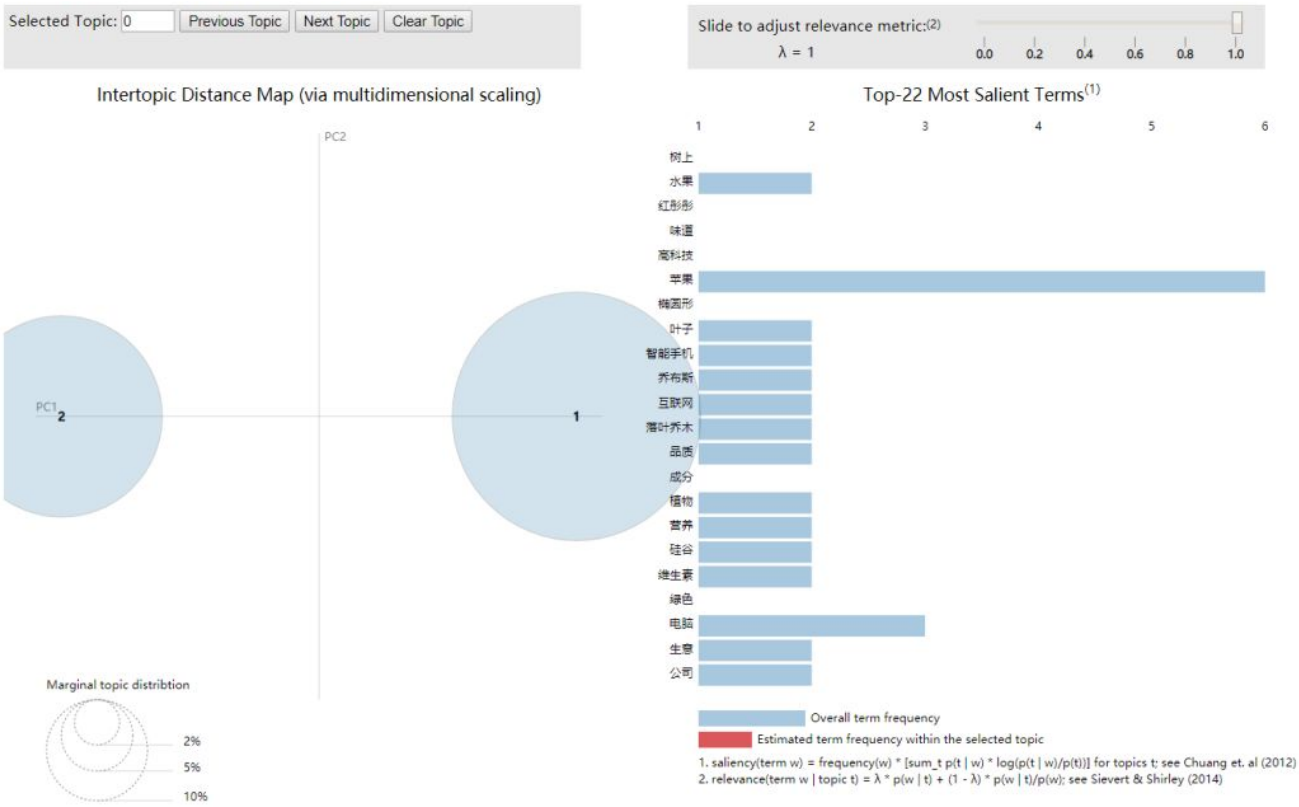
127.0.0.1 - - [05/Jun/2019 14:53:32] "GET / HTTP/1.1" 200 -

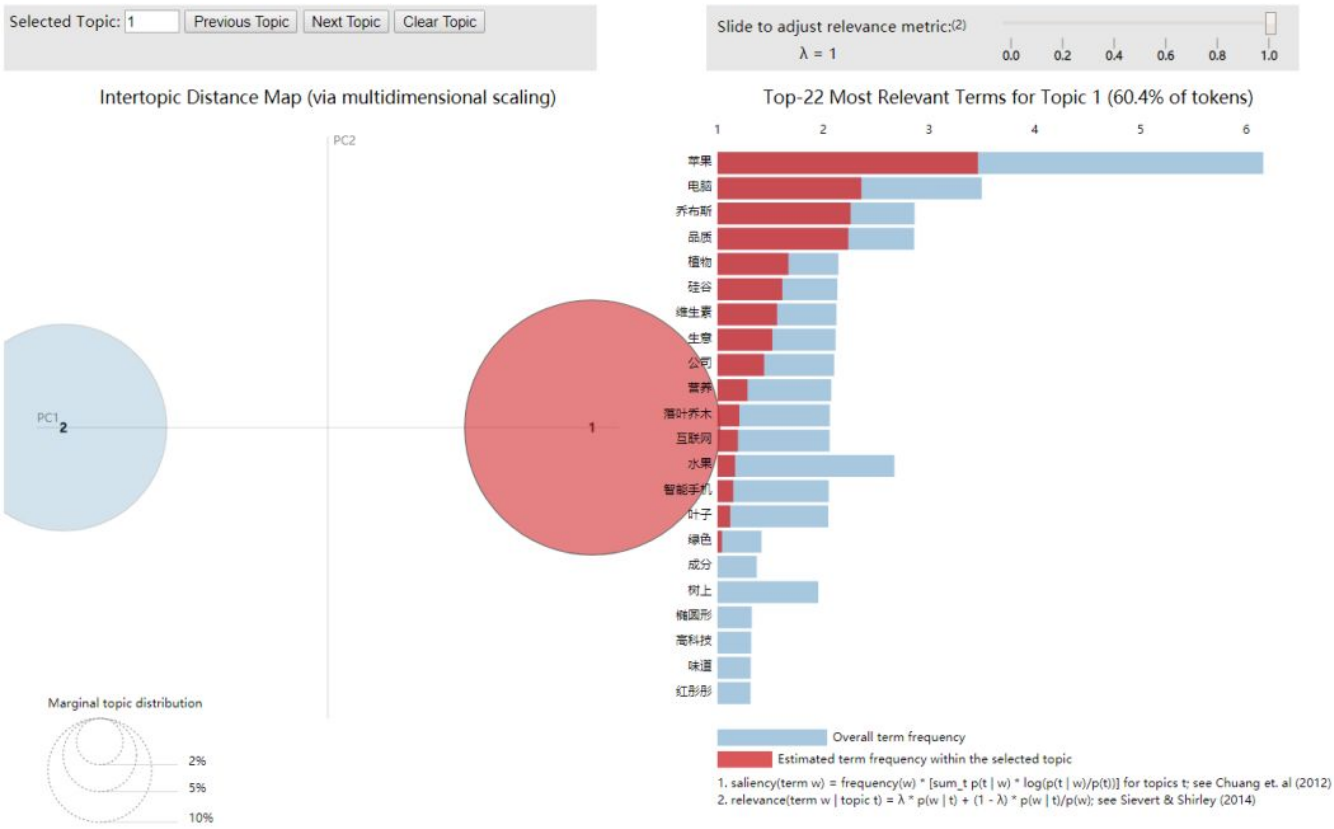
127.0.0.1 - - [05/Jun/2019 14:53:33] "GET /LDAvis.css HTTP/1.1" 200 -

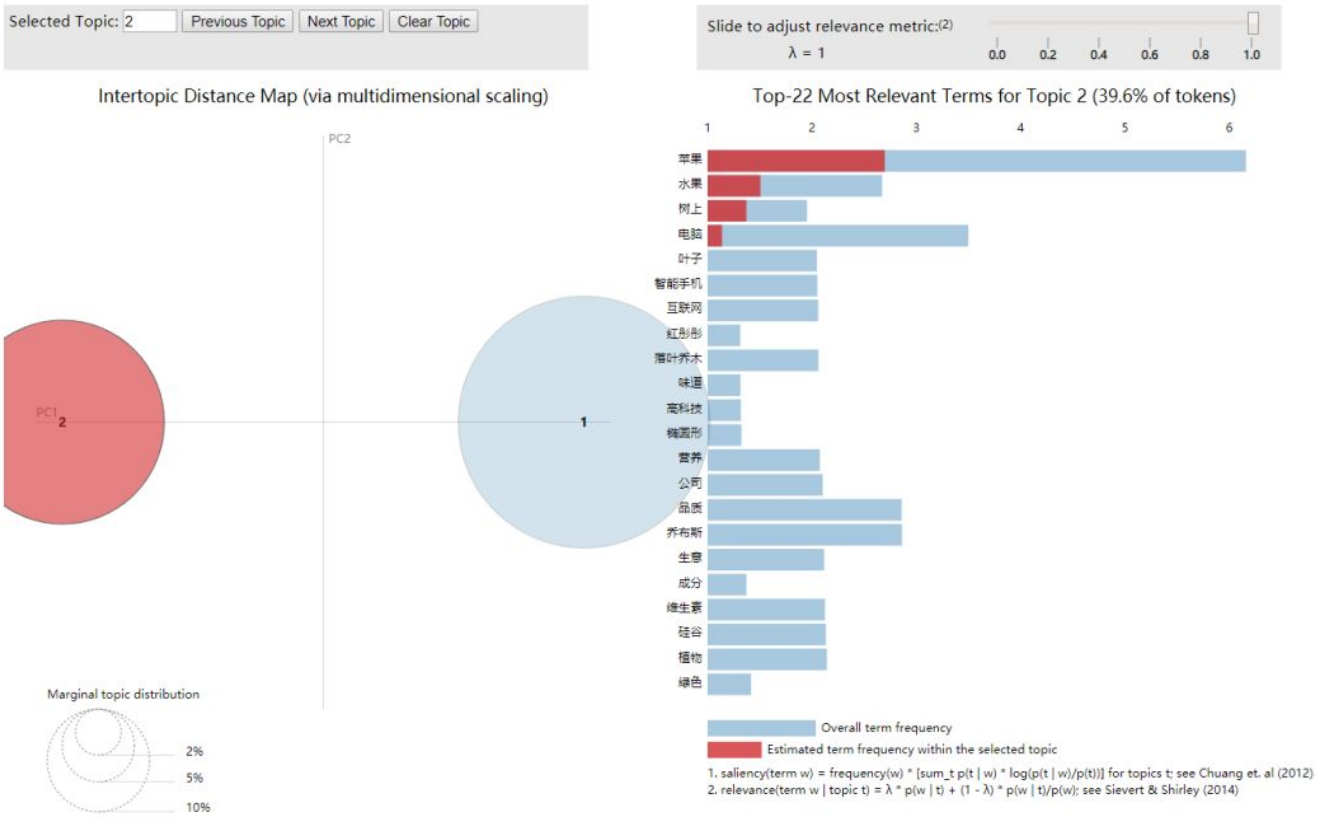
127.0.0.1 - - [05/Jun/2019 14:53:33] "GET /d3.js HTTP/1.1" 200 -

127.0.0.1 - - [05/Jun/2019 14:53:33] "GET /LDAvis.js HTTP/1.1" 200 -

显示完上述日志后，在web浏览器上就会弹出新的标签页，出现可交互的可视化页面。对于可视化结果的解读，笔者不做赘述，可根据上述pyLDAvis着重分析的三个维度来展开。







接下来写点啥呢？笔者目前还没有什么计划，如果你有感兴趣的文本挖掘方向，可以在下面的评论区说说哈，虽然鄙喵不一定都会，但愿闻其详~

推荐阅读

这个NLP工具，玩得根本停不下来

从数据到模型，你可能需要1篇详实的pytorch踩坑指南

如何让Bert在finetune小数据集时更“稳”一点

模型压缩实践系列之——bert-of-theseus，一个非常亲民的bert压缩方法

征稿启示| 200元稿费+5000DBC（价值20个小时GPU算力）

文本自动摘要任务的“不完全”心得总结番外篇——submodular函数优化

Node2Vec 论文+代码笔记

模型压缩实践收尾篇——模型蒸馏以及其他一些技巧实践小结

中文命名实体识别工具（NER）哪家强?

学自然语言处理，其实更应该学好英语

斯坦福大学NLP组Python深度学习自然语言处理工具Stanza试用

关于AINLP

AINLP 是一个有趣有AI的自然语言处理社区，专注于 AI、NLP、机器学习、深度学习、推荐算法等相关技术的分享，主题包括文本摘要、智能问答、聊天机器人、机器翻译、自动生成、知识图谱、预训练模型、推荐系统、计算广告、招聘信息、求职经验分享等，欢迎关注！加技术交流群请添加AINLPer(id：ainlper)，备注工作/研究方向+加群目的。

阅读至此了，分享、点赞、在看三选一吧