文本附加试验1

张配天 2018202180

使用LSI

- set函数构造不重复的iterable集合,split()默认以空格为分隔符
- 使用simple_preprocess处理数据,其会自动将字符串统一为小写,并且转化为列表,设置deacc=Ture来去除特殊符号
- 英文还可以用simple_preprocess来统一不同词性的相同单词
- defaultdict给字典设置缺省值
- *不去除符号的坏影响??*

代码啥的都在下一页

In [122]:

```
from collections import defaultdict
from gensim import corpora
from gensim.utils import simple_preprocess

with open(r'C:\Pt_Python\data\documents_2.txt','r',encoding='utf-8') as f:
    documents = []
    for line in f:
        line = line[:-1]
        documents.append(line)
#print(documents)
def remove_symbol(documents):
    for document in documents:
        yield(simple_preprocess(str(document), deacc=True))

documents = remove_symbol(documents)

stoplist_2 = ['a','of','in','and','to','is','啊','是','的','you','are','will','what','i']
#stoplist = [', ','。',':',' “',"”",':',' ',',','.','/','\'',':','...','-',': ','、','…',',','\"']

for i in range(1,11):
    word = '第'+str(i)+'集'
    stoplist.append(word)
    word = 'E0{}'.format(i)
    stoplist.append(word)
'''
texts = [
    [word for word in document.lower().split() if word not in stoplist]
    for document in documents
]
'''
texts = [
    [word for word in simple_preprocess(str(document)) if word not in stoplist]
    for document in documents
]

frequency = defaultdict(int)
for text in texts:
    for word in text:
        frequency[word] += 1

texts = [
    [word for word in line if frequency[word] >= 2]
    for line in texts
]
```

In [123]:

```
#装载字典
dictionary = corpora.Dictionary(texts)
#生成向量集
corpus = [dictionary.doc2bow(text) for text in texts]
```

In [124]:

```
from gensim import models
#转换成tfidf
tfidf = models.TfidfModel(corpus)
corpus_tfidf = tfidf[corpus]
```

In [125]:

```
lsi_model = models.LsiModel(corpus_tfidf,id2word=dictionary,num_topics=4)
corpus_lsi = lsi_model[corpus_tfidf]
lsi_model.print_topics(4)
```

Out[125]:

```
[(0,
  '-0.503*"最后" + -0.488*"一集" + -0.259*"好看" + -0.256*"第一季" + -0.186*"脑洞" +
-0.177*"第二季" + -0.156*"还是" + -0.131*"悲伤" + -0.130*"rick" + -0.108*"喜欢"'),
 (1,
  '-0.529*"好看" + 0.407*"一集" + 0.402*"最后" + -0.357*"第一季" + -0.217*"everythin
g" + -0.202*"第二季" + -0.200*"for" + -0.113*"还是" + -0.096*"牛逼" + 0.092*"悲
伤"'),
 (2,
  '-0.678*"everything" + -0.633*"for" + 0.247*"好看" + 0.159*"第一季" + 0.089*"第二
季" + -0.069*"催泪" + -0.067*"泪目" + -0.061*"最后" + -0.059*"一集" + -0.058*"lov
e"'),
 (3,
  '-0.649*"好看" + -0.306*"牛逼" + 0.268*"第一季" + 0.255*"脑洞" + -0.173*"一集" + -
0.168*"最后" + 0.164*"第二季" + 0.133*"还是" + 0.131*"宇宙" + 0.110*"喜欢"')]
```

- 效果嘛有待商榷,但是第三个topic做的很好:*rick and morty第二季最后一集,确实相当催泪,也展现了love*
- 但是可以发现第一第二个topic基本类似

使用非负矩阵分解法。

- Dictionary的实例有id2token属性,用于返回以id为索引的单词表,加入这个参数可以在print时输出单词而非id

In [126]:

```
from gensim.models import nmf
id_dict = dictionary.id2token
corpus_nmf = nmf.Nmf(corpus_tfidf,num_topics=3,id2word=id_dict)
result = corpus_nmf.print_topics(3)
result
```

Out[126]:

```
[(0,
  '0.010*"五星" + 0.010*"神作" + 0.010*"宇宙" + 0.010*"第二季" + 0.010*"好看" + 0.01
0*"第三季" + 0.009*"看到" + 0.008*"悲伤" + 0.008*"外公" + 0.008*"结局"'),
 (1,
  '0.019*"一集" + 0.018*"最后" + 0.017*"脑洞" + 0.012*"神剧" + 0.012*"rick" + 0.012
*"编剧" + 0.011*"宇宙" + 0.011*"牛逼" + 0.009*"一个" + 0.009*"瑞克"'),
 (2,
  '0.022*"还是" + 0.013*"everything" + 0.013*"喜欢" + 0.010*"第一季" + 0.009*"hurt"
+ 0.009*"可以" + 0.008*"人类" + 0.008*"竟然" + 0.008*"结尾" + 0.008*"动画"')]
```

In [127]:

```
from gensim.models import Word2Vec
w2v = Word2Vec(texts,min_count=1,size=2)
print(w2v)
```

```
Word2Vec(vocab=742, size=2, alpha=0.025)
```

In [128]:

```
pairs = [
    ('第二季', '好看'),
    ('rick', 'morty')
]
for w1, w2 in pairs:
    print('%r\t%r\t%.2f' % (w1, w2, w2v.wv.similarity(w1, w2)))
```

```
'第二季'        '好看'  0.96
'rick'  'morty' -1.00
```

这里可见

- 第二季确实很好看
- rick确实相当 讨厌 morty了哈哈哈哈哈,你能看到我加双引号的位置么?