案例 5：利用 LDA 算法进行主题词提取

LDA（Latent Dirichlet Allocation）是一种文档主题生成模型，也称为一个三层贝叶斯概率模型，包含词、主题和文档三层结构。所谓生成模型，就是说，我们认为一篇文章的每个词都是通过“以一定概率选择了某个主题，并从这个主题中以一定概率选择某个词语”这样一个过程得到。文档到主题服从多项式分布，主题到词服从多项式分布。

LDA 是一种非监督机器学习技术，可以用来识别大规模文档集（document collection）或语料库（corpus）中潜藏的主题信息。它采用了词袋（bag of words）的方法，这种方法将每一篇文档视为一个词频向量，从而将文本信息转化为了易于建模的数字信息。但是词袋方法没有考虑词与词之间的顺序，这简化了问题的复杂性，同时也为模型的改进提供了契机。每一篇文档代表了一些主题所构成的一个概率分布，而每一个主题又代表了很多单词所构成的一个概率分布。

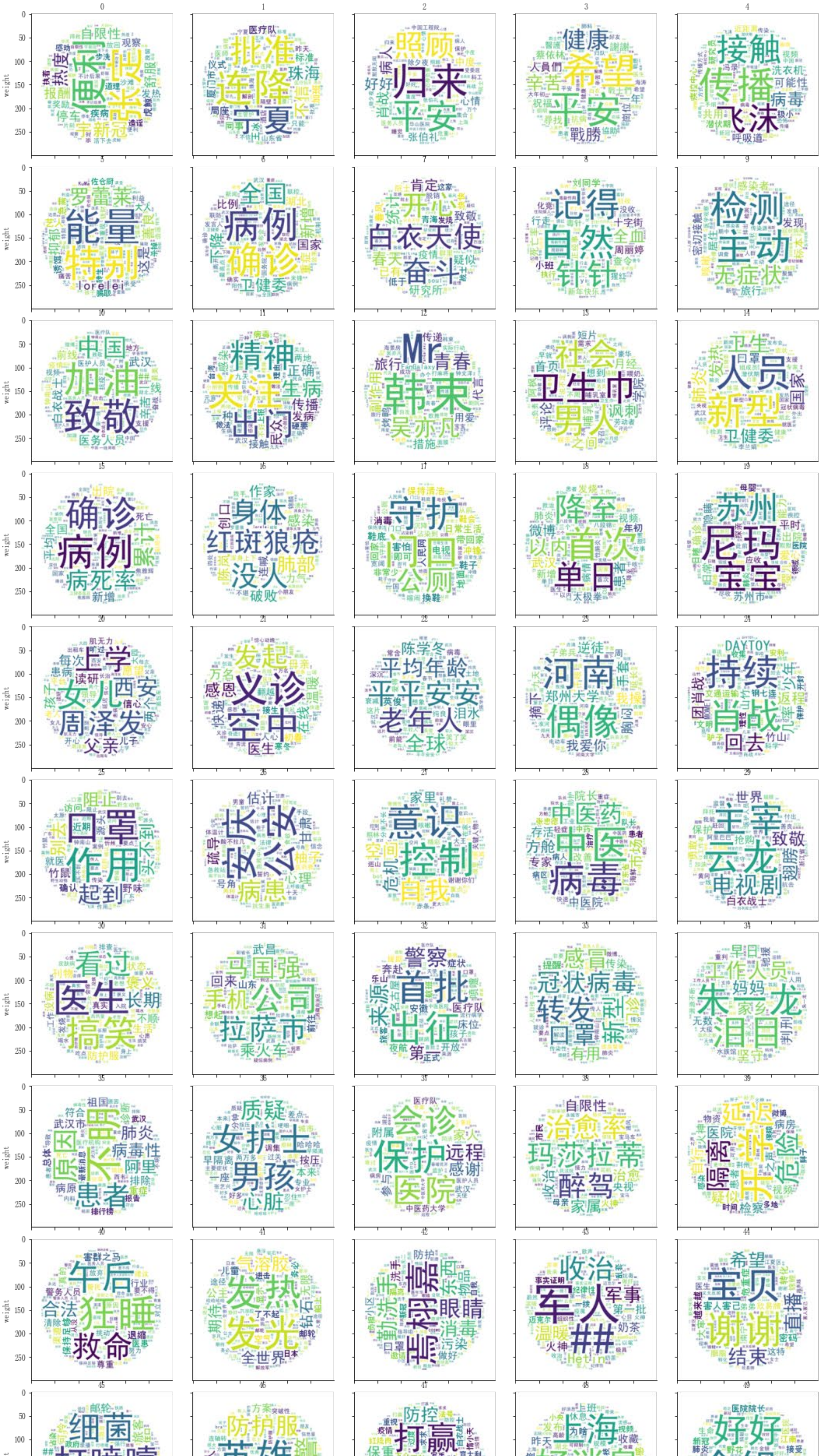
```
# for article in corpus:
dictionary = corpora.Dictionary(corpus)
# 将 dictionary 转化为一个词袋
common_corpus = [dictionary.doc2bow(text) for text in corpus]
tfidf = models.TfidfModel(common_corpus)
corpusTfidf = tfidf[common_corpus]

lda = LdaModel(corpusTfidf, num_topics=50, id2word = dictionary, passes=20)
results = lda.print_topics(num_topics=50, num_words=25)
```

绘制每个主题的分面词云图

```
!pip install wordcloud
import pandas as pd
from wordcloud import WordCloud

res=pd.DataFrame(columns=['topic','word','weight'])
for i in range(len(results)):
    cur_res=results[i][1].split('+')
    cur_res=[x.strip().split('*') for x in cur_res]
    cur_res=[[i,x[1][1:-1],int(float(x[0])*1000)] for x in cur_res]
    cur_res=[x for x in cur_res if x[2]>0]
    cur_res=pd.DataFrame(columns=['topic','word','weight'],data=cur_res)
    if cur_res.shape[0]>3:
        res=res.append(cur_res)
res=res.reset_index(drop=1)

def plot_barplot(*args, **kwargs):
    df = pd.concat([args[0], args[1]], axis=1)
    res=dict(zip(df.iloc[:,0].tolist(),df.iloc[:,1].tolist()))
    x,y = np.ogrid[:300,:300]
```

```
        mask=(x-150) ** 2 + (y-150) ** 2 > 130 ** 2
        masks = 255 * mask.astype(int)
        wordcloud = WordCloud(repeat=True, background_color='white',font_path="
SimHei.ttf",mask = masks)
        wordcloud = wordcloud.generate_from_frequencies(res)
        plt.imshow(wordcloud, interpolation='bilinear')

# 创建分面
g = sns.FacetGrid(res, col='topic', col_wrap=5, margin_titles=True)
g.map(plot_barplot, 'word', 'weight')
g.set_titles(col_template = '{col_name}')

plt.subplots_adjust(top=0.92)
fig = plt.gcf()
fig.suptitle('主题词云图', fontsize=25)
plt.show()
```

主题词云图



weight
0
1
2
3
4
张文宏
便利
批准
连降
宁夏
照顾
归来
平安
希望
平安
健康
接触
传播
飞沫
5
6
7
8
9
能量
特别
病例
确诊
白衣天使
奋斗
开心
自然
记得
针针
检测
主动
无症状
10
11
12
13
14
加油
致敬
中国
精神
关注
出门
韩束
Mr
卫生巾
男人
社会
人员
新型
卫生
15
16
17
18
19
病例
确诊
病死率
红斑狼疮
身体
没人
守护
公厕
降至
首次
单日
尼玛
宝宝
苏州
20
21
22
23
24
女儿
周泽发
上学
义诊
空中
发起
平平安安
老年人
平均年龄
河南
偶像
持续
肖战
25
26
27
28
29
口罩
作用
安庆
公安
意识
控制
自我
中医药
病毒
中医
主宰
云龙
电视剧
30
31
32
33
34
医生
搞笑
看过
公司
拉萨市
马国强
首批
出征
警察
冠状病毒
转发
口罩
朱一龙
泪目
工作人员
35
36
37
38
39
不明
原因
患者
女护士
男孩
质疑
保护
会诊
医院
玛莎拉蒂
醉驾
治愈率
隔离
开学
延迟
40
41
42
43
44
午后
狂睡
救命
发热
发光
气溶胶
焉栩嘉
军人
##
收治
宝贝
谢谢
直播
45
46
47
48
49
细菌
防护服
英雄
防控
打赢
上海
好好

- 基于 pyLDAvis 的主题模型可视化

```
#lda: 计算好的话题模型;corpus: 文档词频矩阵;dictionary: 词语空间
import pyLDAvis.gensim_models
d=pyLDAvis.gensim_models.prepare(lda, corpusTfidf, dictionary)
pyLDAvis.save_html(d, './result/lda.html')  # 将结果保存为该 html 文件
```

绘制一段话的主题词云图

```
# 将 LDA 模型保存起来方便日后对新文章进行主题预测

test = "武汉，华科同济医学院协和医院 7 名医护人员经过治疗，初步检测结果显示，新型冠状病毒核酸检测已转阴，临床症状得到控制。一名医护人员视频通话报了平安“不发烧了，精神很好”，隔着隔离室的玻璃比了 ok 手势"

test = jieba.lcut(test)

# 文档转换成 bow
doc_bow = dictionary.doc2bow(test)
# 得到新文档的主题分布
doc_lda = lda[doc_bow]
print(doc_lda)
[(1, 0.035826407), (3, 0.069355994), (9, 0.06360182), (14,
 0.03542321), (16, 0.044641234), (20, 0.104288325), (28,
0.034048904), (36, 0.16313529), (41, 0.32614222), (48, 0.0
9676166)]
type = doc_lda[4][0]
print(results[type][1])
0.030*"辛苦" + 0.020*"健康" + 0.019*"希望" + 0.014*"平安" +

0.010*"戰勝" + 0.010*"人員們" + 0.010*"潜伏期" + 0.010*"一年"

+ 0.010*"蔡依林" + 0.010*"祝福" + 0.009*"謝謝" + 0.009*"協助"

 + 0.009*"尋找" + 0.009*"崗位" + 0.009*"戰士們" + 0.009*"抗病"

 + 0.009*"醫護" + 0.009*"大年初一" + 0.008*"确实" + 0.008*"白

衣" + 0.006*"呼吸衰竭" + 0.006*"患者" + 0.005*"央视" + 0.005*"

高占成" + 0.005*"休克"
from wordcloud import WordCloud

res=results[type][1].split('+')
```

```
res=[x.strip().split('*') for x in res]
res=[[int(float(x[0])*1000),x[1][1:-1]] for x in res]
res=np.array([x for x in res if x[0]>0])
weight=res[:,0].astype(np.int16)
word=res[:,1]
res=dict(zip(word,weight))

wordcloud = WordCloud(width=500, height=500, repeat=True, background_color
='white', font_path="msyh.ttc")
wordcloud = wordcloud.generate_from_frequencies(res)

plt.figure(figsize=(8, 8))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```