**Appendix III**

**Code for Topic identify and topic classification**

import pandas as pd

df = pd.read\_excel('./csv文件/excel文件/整体数据.xlsx')

df

import re

import jieba

stoptext = open('./csv文件/中文.txt', encoding='utf-8').read()

stopwords = stoptext.split('\n')

def clean\_text(text):

words = jieba.lcut(text)

words = [w for w in words if w not in stopwords]

return ' '.join(words)

df['content'] = df['评价'].apply(clean\_text)

df.head()

from sklearn.feature\_extraction.text import CountVectorizer,TfidfVectorizer

vectorizer = CountVectorizer(max\_df=0.5,

min\_df=20)

doc\_term\_matrix = vectorizer.fit\_transform(df['content'])

doc\_term\_matrix

from sklearn.decomposition import LatentDirichletAllocation

# 构建LDA话题模型

lda\_model = LatentDirichletAllocation(n\_components=5) # 话题数

lda\_output = lda\_model.fit\_transform(doc\_term\_matrix)

print(lda\_model) #模型参数

print(lda\_output) #话题分布情况

vectorizer.get\_feature\_names()

def show\_topics(vectorizer, lda\_model, top\_n=20):

import numpy as np

keywords = np.array(vectorizer.get\_feature\_names())

topic\_keywords = []

#话题-词语权重矩阵

for topic\_weights in lda\_model.components\_:

#获得权重最大的top\_n词语的权重向量

top\_keyword\_locs = (-topic\_weights).argsort()[:top\_n]

#在keywords中找到对于的关键词

topic\_keywords.append(keywords.take(top\_keyword\_locs))

return topic\_keywords

topic\_keywords = show\_topics(vectorizer= vectorizer,

lda\_model = lda\_model,

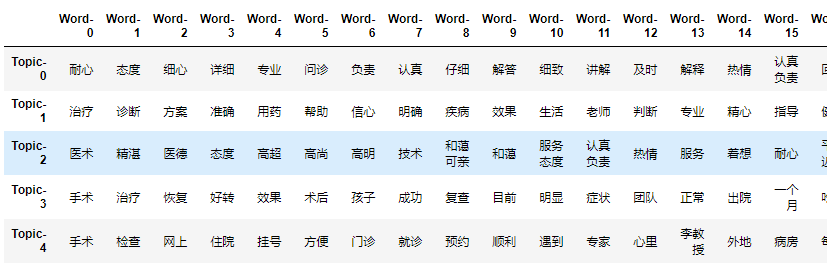
top\_n = 20)

df\_topic\_keywords = pd.DataFrame(topic\_keywords)

df\_topic\_keywords.columns = ['Word-'+str(i) for i in range(df\_topic\_keywords.shape[1])]

df\_topic\_keywords.index = ['Topic-'+str(i) for i in range(df\_topic\_keywords.shape[0])]

df\_topic\_keywords



**Topic Extraction Run Result**

import numpy as np

# cntVector = CountVectorizer()

# cntTf = cntVector.fit\_transform(cmtlst)

# 构建文档-词频矩阵

lda\_output = lda\_model.transform(doc\_term\_matrix)

# 列名

topicnames = ["Topic" + str(i)

for i in range(lda\_model.n\_components)]

docnames = ["Doc" + str(i)

for i in range(len(df['评价']))]

# 转化为pd.DataFrame

df\_document\_topic = pd.DataFrame(np.round(lda\_output, 2),

columns=topicnames,

index=docnames)

# Get dominant topic for each document

dominant\_topic = np.argmax(df\_document\_topic.values, axis=1)

df\_document\_topic['dominant\_topic'] = dominant\_topic

# Styling

def color\_green(val):

color = 'green' if val > .1 else 'black'

return 'color: {col}'.format(col=color)

def make\_bold(val):

weight = 700 if val > .1 else 400

return 'font-weight: {weight}'.format(weight=weight)

# Apply Style

df\_document\_topics = df\_document\_topic.sample(10).style.applymap(color\_green).applymap(make\_bold)

df\_document\_topics



**Reviewss-topics classification Run Result**

df\_topic\_distribution = df\_document\_topic['dominant\_topic'].value\_counts().reset\_index(name="Num Documents")

df\_topic\_distribution.columns = ['Topic Num', 'Num Documents']

df\_topic\_distribution

df\_document\_topic.reset\_index(drop=False,inplace=True)

df\_document\_topic

cc=pd.concat([df\_document\_topic,df], ignore\_index=True,axis=1)

cc.to\_excel('./csv文件/excel文件/预测数据.xlsx') #合并保存