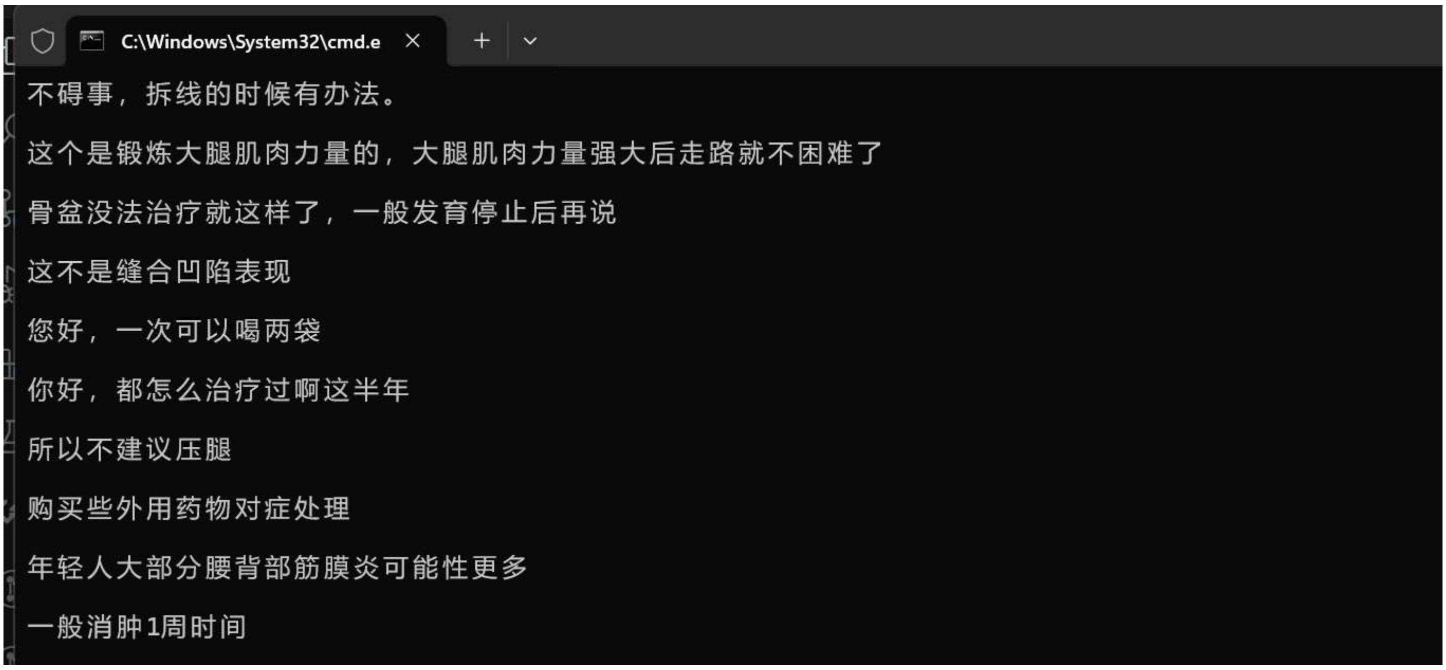
词频统计

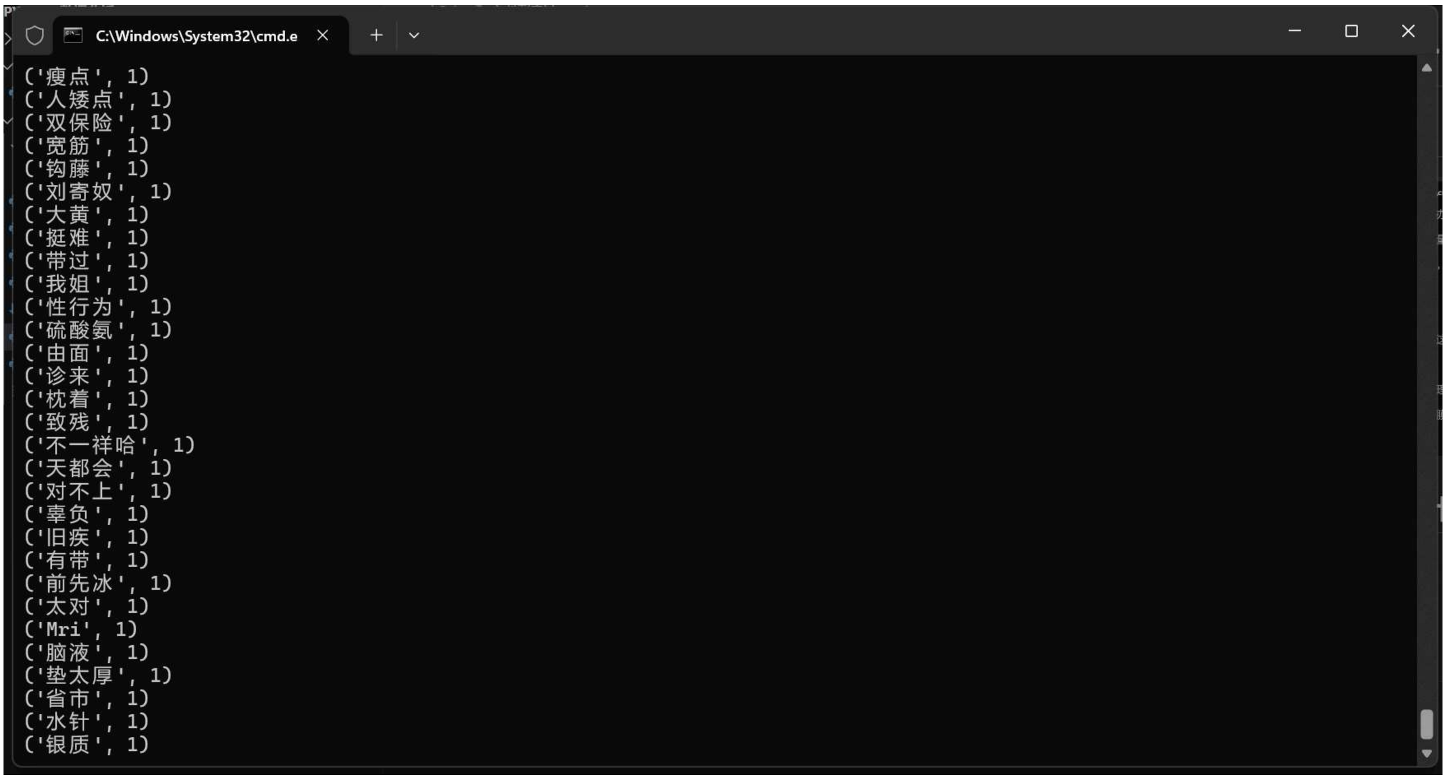
1.读取文档，打印前10行



```
不碍事，拆线的时候有办法。
这个是锻炼大腿肌肉力量的，大腿肌肉力量强大后走路就不困难了
骨盆没法治疗就这样了，一般发育停止后再说
这不是缝合凹陷表现
您好，一次可以喝两袋
你好，都怎么治疗过啊这半年
所以不建议压腿
购买些外用药物对症处理
年轻人大部分腰背部筋膜炎可能性更多
一般消肿1周时间
```

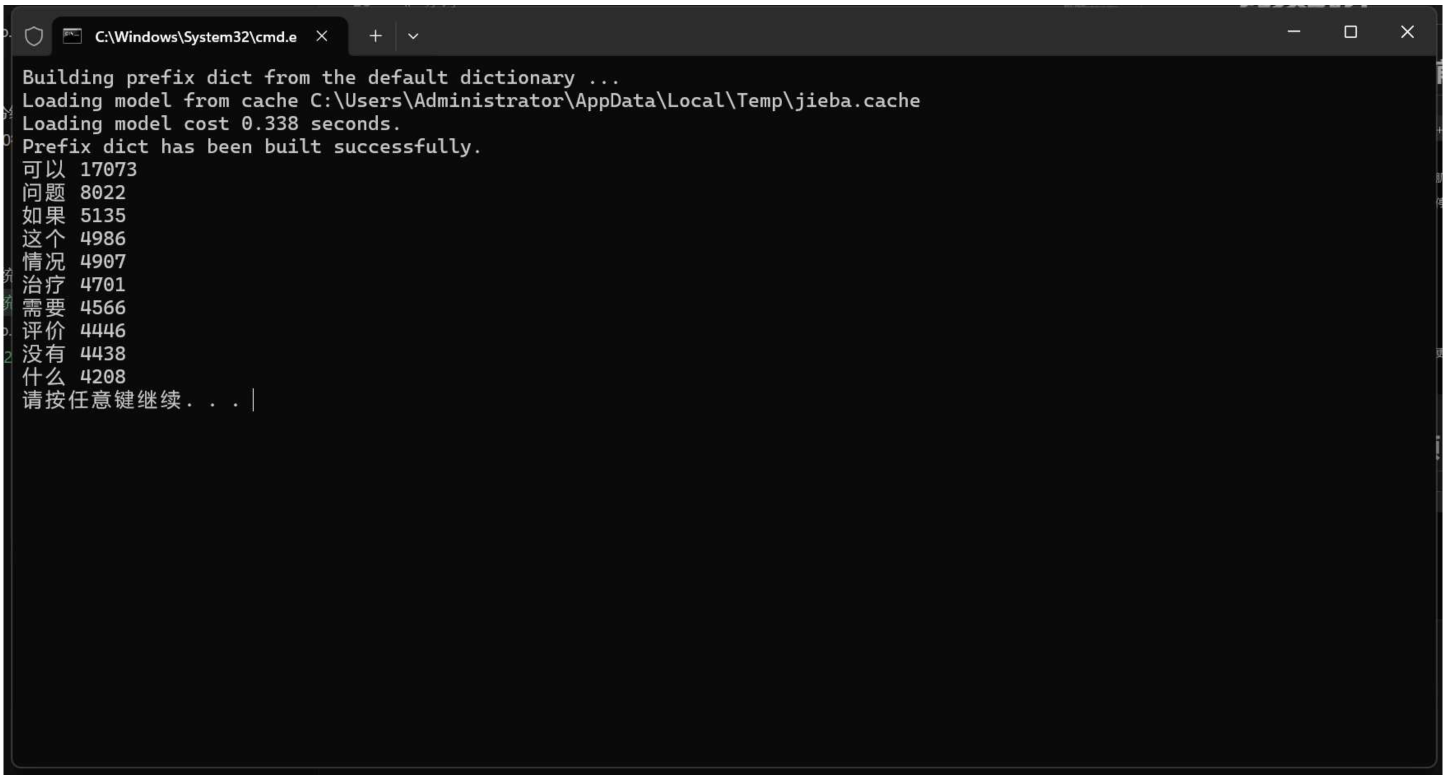
2.分词并统计词频

一部分结果



```
('瘦点', 1)
('人矮点', 1)
('双保险', 1)
('宽筋', 1)
('钩藤', 1)
('刘寄奴', 1)
('大黄', 1)
('挺难', 1)
('带过', 1)
('我姐', 1)
('性行为', 1)
('硫酸氨', 1)
('由面', 1)
('诊来', 1)
('枕着', 1)
('致残', 1)
('不一样哈', 1)
('天都会', 1)
('对不上', 1)
('辜负', 1)
('旧疾', 1)
('有带', 1)
('前先冰', 1)
('太对', 1)
('Mri', 1)
('脑液', 1)
('垫太厚', 1)
('省市', 1)
('水针', 1)
('银质', 1)
```

3.词频最高的10个词

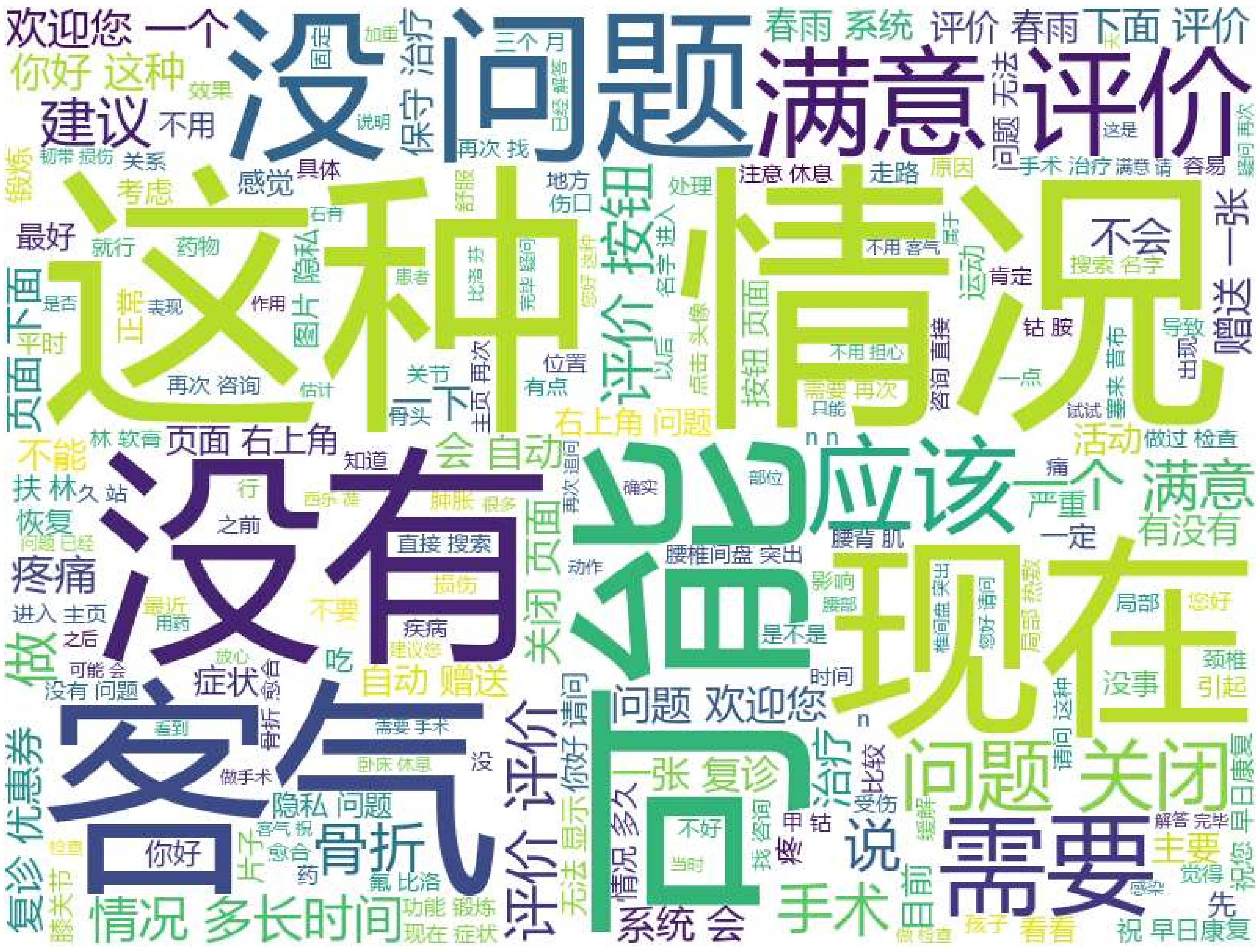


```
Building prefix dict from the default dictionary ...
Loading model from cache C:\Users\Administrator\AppData\Local\Temp\jieba.cache
Loading model cost 0.338 seconds.
Prefix dict has been built successfully.
可以 17073
问题 8022
如果 5135
这个 4986
情况 4907
治疗 4701
需要 4566
评价 4446
没有 4438
什么 4208
请按任意键继续. . .
```

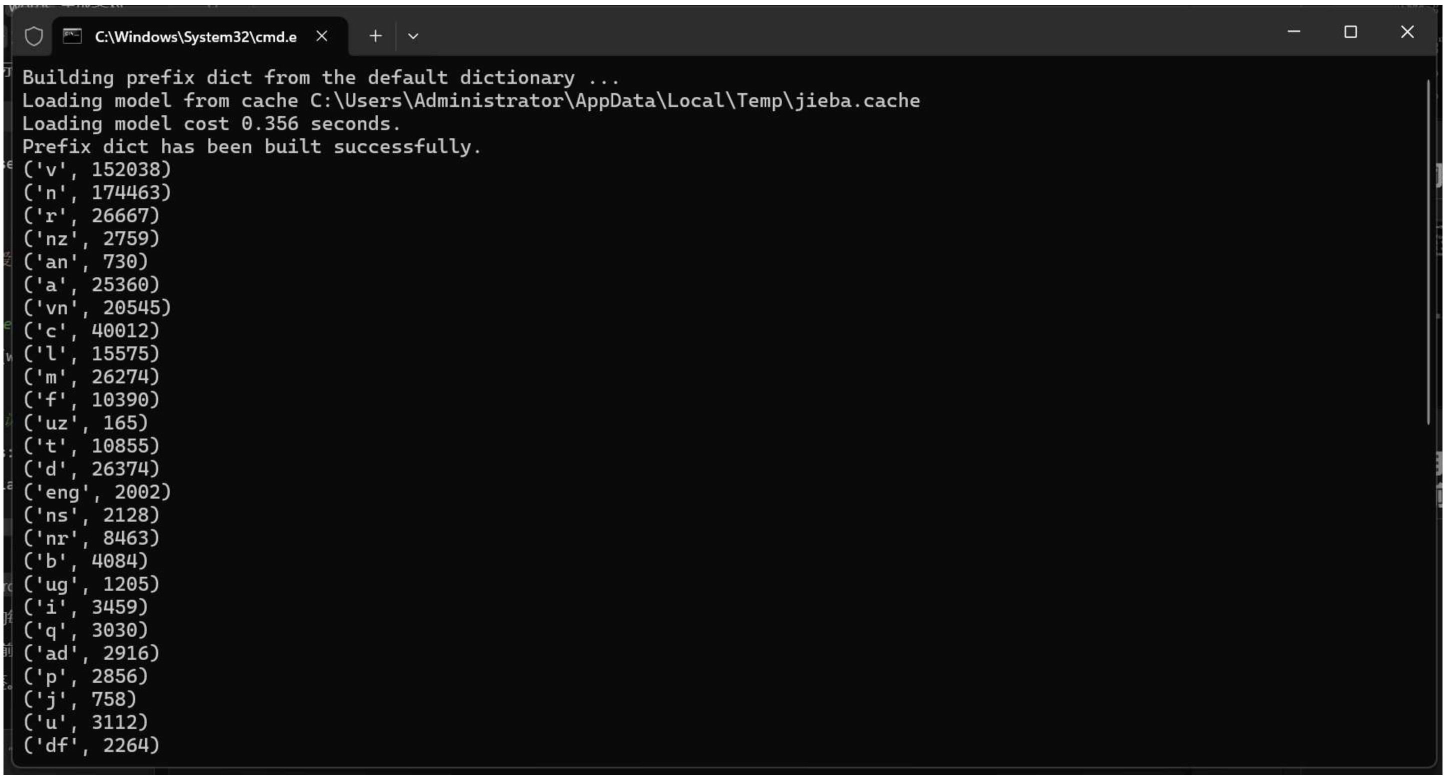
4 5.用词云可视化，引入停用词表前后对比

- 未加入停用词表

- 加入停用词表

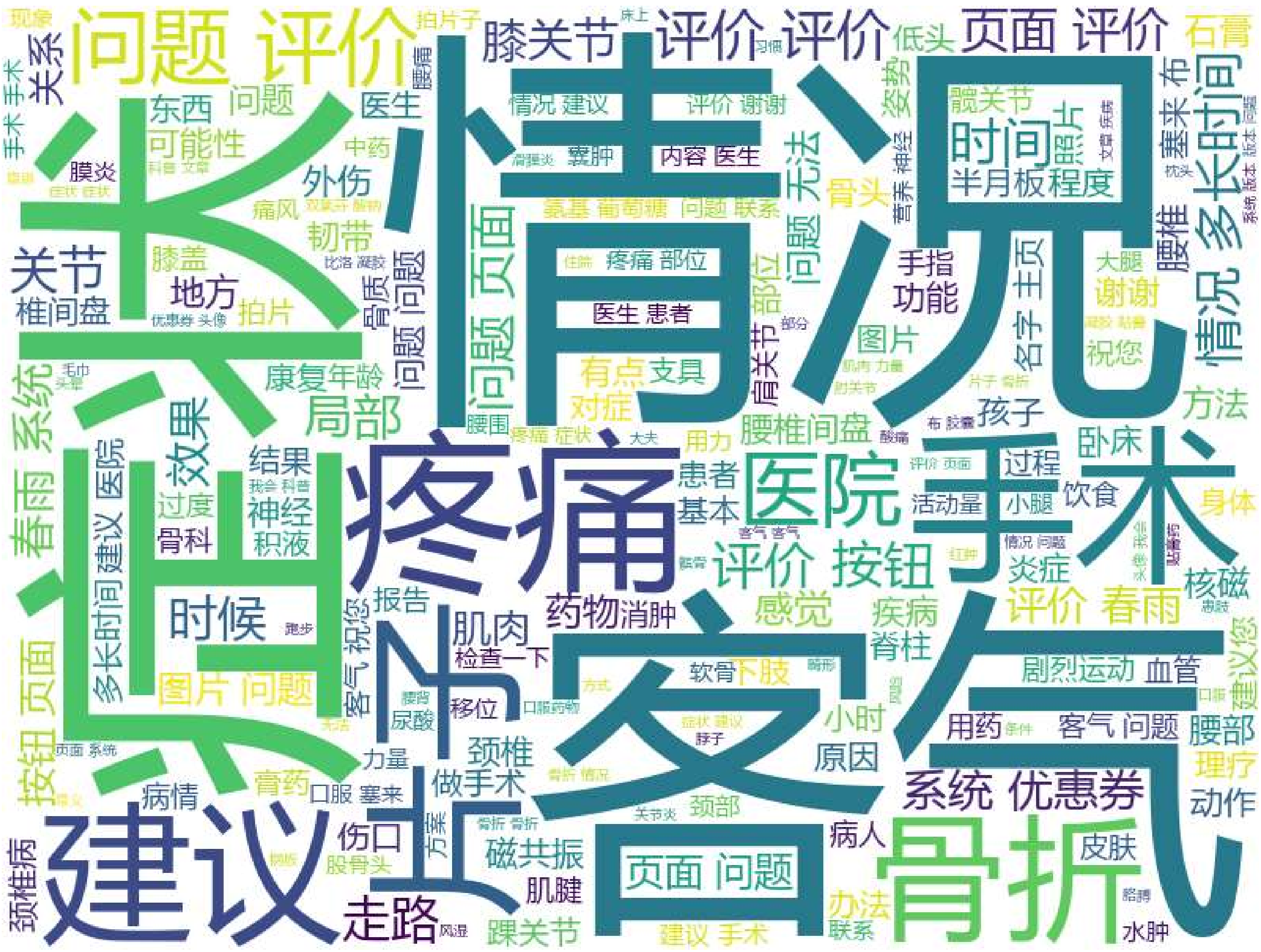


6.词性统计

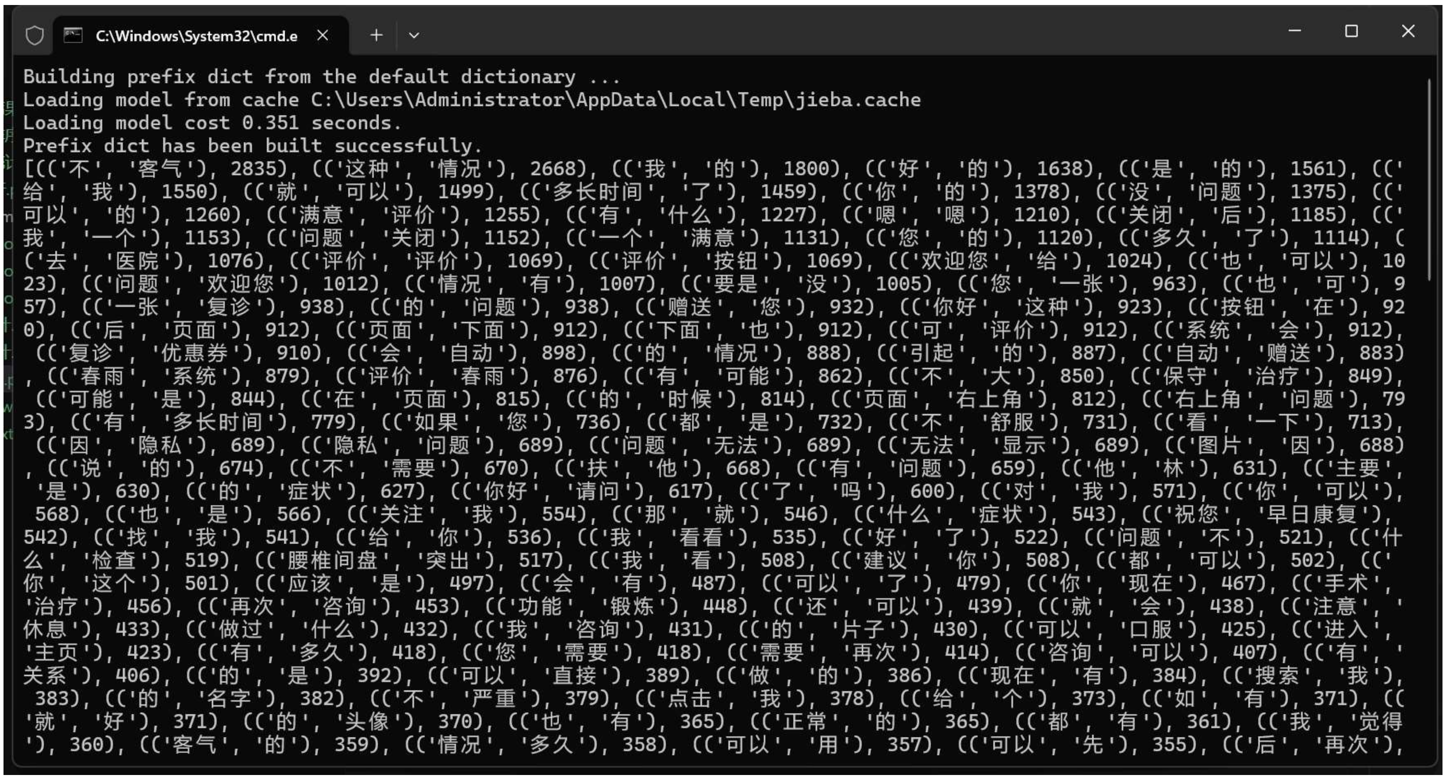


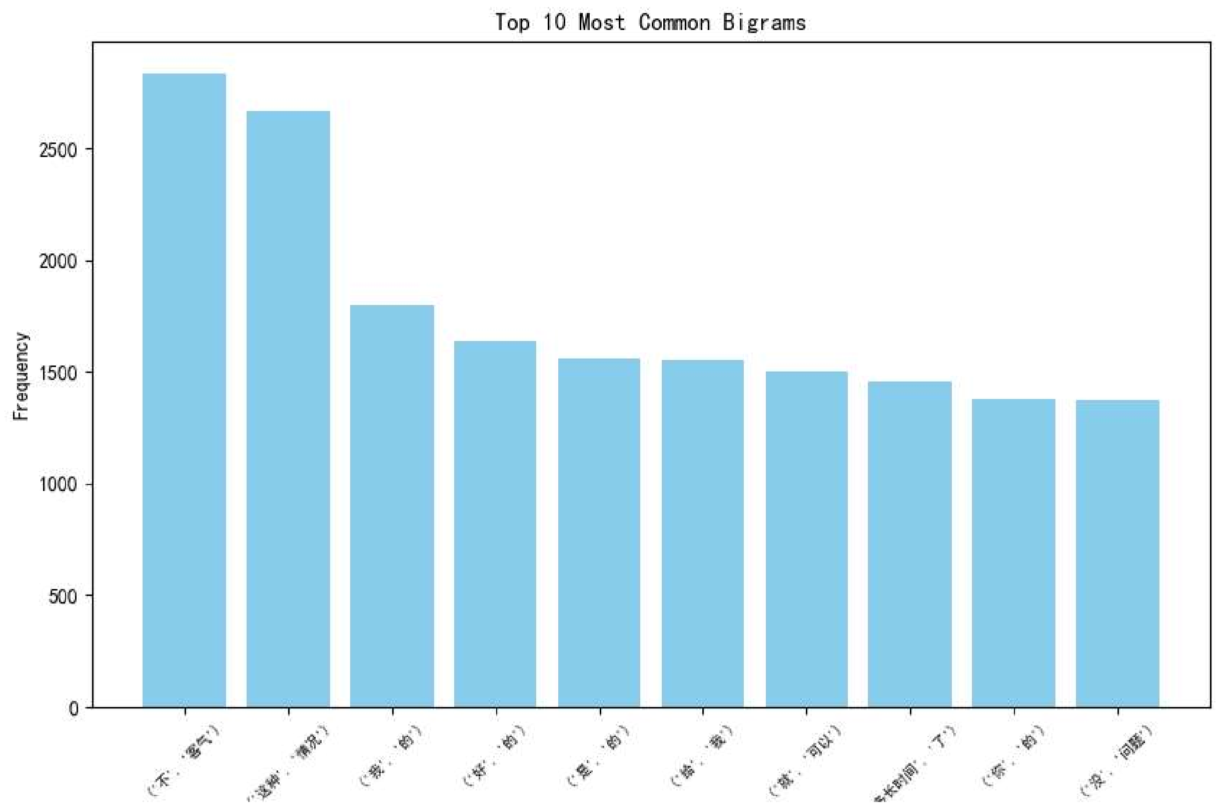
```
Building prefix dict from the default dictionary ...
Loading model from cache C:\Users\Administrator\AppData\Local\Temp\jieba.cache
Loading model cost 0.356 seconds.
Prefix dict has been built successfully.
('v', 152038)
('n', 174463)
('r', 26667)
('nz', 2759)
('an', 730)
('a', 25360)
('vn', 20545)
('c', 40012)
('l', 15575)
('m', 26274)
('f', 10390)
('uz', 165)
('t', 10855)
('d', 26374)
('eng', 2002)
('ns', 2128)
('nr', 8463)
('b', 4084)
('ug', 1205)
('i', 3459)
('q', 3030)
('ad', 2916)
('p', 2856)
('j', 758)
('u', 3112)
('df', 2264)
```

对名词进行词云可视化



7.统计bi-gram频率并可视化高频词


```
Building prefix dict from the default dictionary ...
Loading model from cache C:\Users\Administrator\AppData\Local\Temp\jieba.cache
Loading model cost 0.351 seconds.
Prefix dict has been built successfully.
[(('不', '客气'), 2835), (('这种', '情况'), 2668), (('我', '的'), 1800), (('好', '的'), 1638), (('是', '的'), 1561), (('
给', '我'), 1550), (('就', '可以'), 1499), (('多长时间', '了'), 1459), (('你', '的'), 1378), (('没', '问题'), 1375), (('
可以', '的'), 1260), (('满意', '评价'), 1255), (('有', '什么'), 1227), (('嗯', '嗯'), 1210), (('关闭', '后'), 1185), (('
我', '一个'), 1153), (('问题', '关闭'), 1152), (('一个', '满意'), 1131), (('您', '的'), 1120), (('多久', '了'), 1114), (
('去', '医院'), 1076), (('评价', '评价'), 1069), (('评价', '按钮'), 1069), (('欢迎您', '给'), 1024), (('也', '可以'), 10
23), (('问题', '欢迎您'), 1012), (('情况', '有'), 1007), (('要是', '没'), 1005), (('您', '一张'), 963), (('也', '可'), 9
57), (('一张', '复诊'), 938), (('的', '问题'), 938), (('赠送', '您'), 932), (('你好', '这种'), 923), (('按钮', '在'), 92
0), (('后', '页面'), 912), (('页面', '下面'), 912), (('下面', '也'), 912), (('可', '评价'), 912), (('系统', '会'), 912),
 (('复诊', '优惠券'), 910), (('会', '自动'), 898), (('的', '情况'), 888), (('引起', '的'), 887), (('自动', '赠送'), 883)
, (('春雨', '系统'), 879), (('评价', '春雨'), 876), (('有', '可能'), 862), (('不', '大'), 850), (('保守', '治疗'), 849),
 (('可能', '是'), 844), (('在', '页面'), 815), (('的', '时候'), 814), (('页面', '右上角'), 812), (('右上角', '问题'), 79
3), (('有', '多长时间'), 779), (('如果', '您'), 736), (('都', '是'), 732), (('不', '舒服'), 731), (('看', '一下'), 713),
 (('因', '隐私'), 689), (('隐私', '问题'), 689), (('问题', '无法'), 689), (('无法', '显示'), 689), (('图片', '因'), 688)
, (('说', '的'), 674), (('不', '需要'), 670), (('扶', '他'), 668), (('有', '问题'), 659), (('他', '林'), 631), (('主要',
 '是'), 630), (('的', '症状'), 627), (('你好', '请问'), 617), (('了', '吗'), 600), (('对', '我'), 571), (('你', '可以'),
 568), (('也', '是'), 566), (('关注', '我'), 554), (('那', '就'), 546), (('什么', '症状'), 543), (('祝您', '早日康复'),
542), (('找', '我'), 541), (('给', '你'), 536), (('我', '看看'), 535), (('好', '了'), 522), (('问题', '不'), 521), (('什
么', '检查'), 519), (('腰椎间盘', '突出'), 517), (('我', '看'), 508), (('建议', '你'), 508), (('都', '可以'), 502), (('
你', '这个'), 501), (('应该', '是'), 497), (('会', '有'), 487), (('可以', '了'), 479), (('你', '现在'), 467), (('手术',
'治疗'), 456), (('再次', '咨询'), 453), (('功能', '锻炼'), 448), (('还', '可以'), 439), (('就', '会'), 438), (('注意', '
休息'), 433), (('做过', '什么'), 432), (('我', '咨询'), 431), (('的', '片子'), 430), (('可以', '口服'), 425), (('进入',
'主页'), 423), (('有', '多久'), 418), (('您', '需要'), 418), (('需要', '再次'), 414), (('咨询', '可以'), 407), (('有', '
关系'), 406), (('的', '是'), 392), (('可以', '直接'), 389), (('做', '的'), 386), (('现在', '有'), 384), (('搜索', '我'),
 383), (('的', '名字'), 382), (('不', '严重'), 379), (('点击', '我'), 378), (('给', '个'), 373), (('如', '有'), 371), ((
'就', '好'), 371), (('的', '头像'), 370), (('也', '有'), 365), (('正常', '的'), 365), (('都', '有'), 361), (('我', '觉得
'), 360), (('客气', '的'), 359), (('情况', '多久'), 358), (('可以', '用'), 357), (('可以', '先'), 355), (('后', '再次'),
```

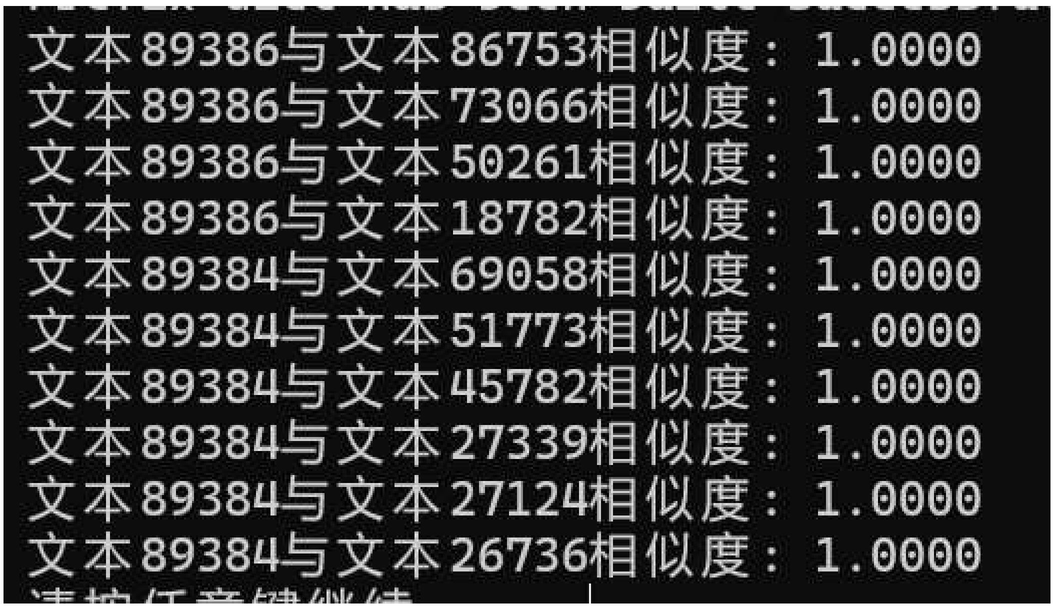


8. 特征词和相似度计算

根据词频排序，选择前2500个词作为特征词

选用词袋向量的方法对每一行文本进行向量表示

经过一系列优化后，通过近似最近邻搜索方法找到相似度最高10条文本结果如下


```
文本89386与文本86753相似度: 1.0000
文本89386与文本73066相似度: 1.0000
文本89386与文本50261相似度: 1.0000
文本89386与文本18782相似度: 1.0000
文本89384与文本69058相似度: 1.0000
文本89384与文本51773相似度: 1.0000
文本89384与文本45782相似度: 1.0000
文本89384与文本27339相似度: 1.0000
文本89384与文本27124相似度: 1.0000
文本89384与文本26736相似度: 1.0000
请按任意键继续
```

代码

词频统计

1.

```
import jieba
import wordcloud
# 读取和分词
with open(r'week2\词频\week2.txt', 'r', encoding='utf-8') as f:
    file = f.read()
    # file_lis = f.readlines()
    # for i in range(10):
    #     print(file_lis[i])

# 分词
words = jieba.lcut(file)

# 统计词频
counts = {}

for word in words:
    if len(word) == 1:
        continue
    else:
        counts[word] = counts.get(word, 0) + 1

# print(*counts.items(),sep='\n')

# 转化为列表进行后续处理
counts_lis = counts.items()
counts_lis = sorted(counts_lis, key=lambda x: x[1], reverse=True)

# 输出前10个高频词
def show_top10():
    for i in range(10):
        word, count = counts_lis[i][0], counts_lis[i][1]
        print(f'{word} {count}')

# 处理为词云可以处理的格式
words_show_root = [word for word in words if len(word) > 1]
words_show = ' '.join(words_show_root)
```

```
def wordcloud_show():
    # 生成词云
    wc = wordcloud.WordCloud(font_path='msyh.ttc', width=800, height=600, backgr
    wc.generate(words_show)
    wc.to_file(r'week2\词频\img\wordcloud.png')



def wordcloud_show_stop():
    # 引入停用词表
    with open(r'week2\词频\cn_stopwords.txt', 'r', encoding='utf-8') as f:
        stopwords = f.read().splitlines()

    words_show_stop = ' '.join([word for word in words if word not in stopwords]

    # 再次生成词云
    wc = wordcloud.WordCloud(font_path='msyh.ttc', width=800, height=600, backgr
    wc.generate(words_show_stop)
    wc.to_file(r'week2\词频\img\wordcloud_stop.png')

# 词性分析
def posseg_show():
    import jieba.posseg as pseg
    words = pseg.cut(''.join(words_show_root))
    posseg_count = {}

    # # 统计词性频率
    # for _, flag in words:
    #     posseg_count[flag] = posseg_count.get(flag, 0) + 1
    # print(*posseg_count.items(),sep='\n')

    # 对名词词性进行词云可视化
    words_show_posseg = ' '.join([word for word, flag in words if flag[0] == 'n'
    wc = wordcloud.WordCloud(font_path='msyh.ttc', width=800, height=600, backgr
    wc.generate(words_show_posseg)
    wc.to_file(r'week2\词频\img\wordcloud_n.png')

# 利用词频筛选特征词

def feature_words():
    # word_show_root是初始文本，处理停用词
    with open(r'week2\词频\cn_stopwords.txt', 'r', encoding='utf-8') as f:
        stopwords = f.read().splitlines()
    words_show_stop = [word for word in words_show_root if word not in stopwords

    # 进行词频统计
```

```
counts = {}
for word in words_show_stop:
    counts[word] = counts.get(word, 0) + 1

# 转化为列表进行后续处理
counts_lis = counts.items()
counts_lis = sorted(counts_lis, key=lambda x: x[1], reverse=True)
# print(counts_lis)

# 保留前2500个词作为特征词
feature_words = [item[0] for item in counts_lis[:2500]]
# print(feature_words)

from sklearn.neighbors import NearestNeighbors  # 替代LSHForest
from sklearn.feature_extraction.text import CountVectorizer
import jieba

feature_words = [word.lower() for word in feature_words]  # 统一转小写
feature_words = list(set(feature_words))

vectorizer = CountVectorizer(
    vocabulary=feature_words,
    lowercase=True
)

with open(r'week2\词频\week2.txt', 'r', encoding='utf-8') as f:
    corpus = [line.strip() for line in f.readlines()]

corpus = [' '.join(jieba.lcut(line)) for line in corpus]


bow_matrix = vectorizer.fit_transform(corpus)

import numpy as np

# 只考虑非零向量
row_norms = np.sqrt((bow_matrix.power(2)).sum(axis=1))
nonzero_mask = np.array(row_norms).flatten() > 0

bow_matrix = bow_matrix[nonzero_mask]


# 近似最近邻搜索
nbrs = NearestNeighbors(n_neighbors=10, algorithm='auto', metric='cosine')
nbrs.fit(bow_matrix)
distances, indices = nbrs.kneighbors(bow_matrix)
```

```
    # 提取top10
    top_pairs = []
    for i, neighbors in enumerate(indices):
        for j, idx in enumerate(neighbors):
            if i != idx:
                top_pairs.append( (1 - distances[i][j], i, idx) )
    top_pairs = sorted(list(set(top_pairs)), reverse=True)[:10]

    # 打印结果
    for sim, i, j in top_pairs:
        print(f"文本{i}与文本{j}相似度: {sim:.4f}")
if __name__ == '__main__':
    # show_top10()
    # wordcloud_show_stop()
    # wordcloud_show()
    # posseg_show()
    feature_words()
```

2.

```python
import nltk
import jieba
import re
import matplotlib.pyplot as plt

with open(r'week2\词频\week2.txt', 'r', encoding='utf-8') as f:
    file = f.readlines()

bi_gram = []
# 按行进行分词和二元组提取
for txt in file:
    txt = re.sub(r'[^\w\s]', '', txt[:-1])
    txt = re.sub(r' ', '', txt)
    tokens = jieba.lcut(txt)
    bi_gram.extend(list(nltk.bigrams(tokens)))

# 统计二元组出现的频率
bi_gram_freq = nltk.FreqDist(bi_gram)
print(bi_gram_freq.most_common())

top_10 = bi_gram_freq.most_common(10)

# 分解二元组和频率
bigrams, frequencies = zip(*top_10)
x_list = [i for i in range(10)]
# 绘制条形图
plt.figure(figsize=(10, 6))
plt.bar(x_list, frequencies, color='skyblue')
plt.xlabel('Bigrams')
plt.ylabel('Frequency')
plt.title('Top 10 Most Common Bigrams')
plt.xticks(x_list,bigrams, rotation=45,fontsize=7)
plt.savefig(r'week2\词频\img\bi_gram.png')
plt.show()
```