# 数据分析

## 数据获取到的第一步就是数据处理工作

这里首先是把获取的数据采用pandas中的read_excel来读取数据表格，然后去进行去重drop_duplicates，再把多余的空值进行删除

接着我们调用os库去创建不同大学的文件夹，这样做的目的是为了把不同分析的文件区分出来，也是为了干净整洁

```
    if not os.path.exists("./重庆大学"):
        os.mkdir("./重庆大学")
```

## 接着进行数据分析工作，先进行NLP以及词云分析

首先我们进行的是词云处理

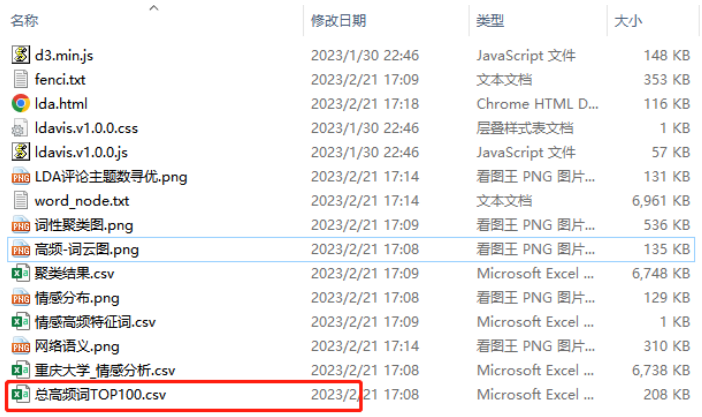
这里我们首先需要一个停用词表，停用词表的意义就是帮助我们筛除一些无意义的词，也就是无效词和标点符号


接着我们根据jieba进行分词处理，分好词之后，我们写一个中文函数，也就是判断这个词是否为中文，如果是中文就保留下来，不是的话就去除，这个便是中文判断函数

```
def is_all_chinese(strs):
    for _char in strs:
        if not '\u4e00' <= _char <= '\u9fa5':
            return False
    return True
```

把词处理好之后，这里我们采用的stylecloud这个词云库，帮助我们有效的将词处理成词云图

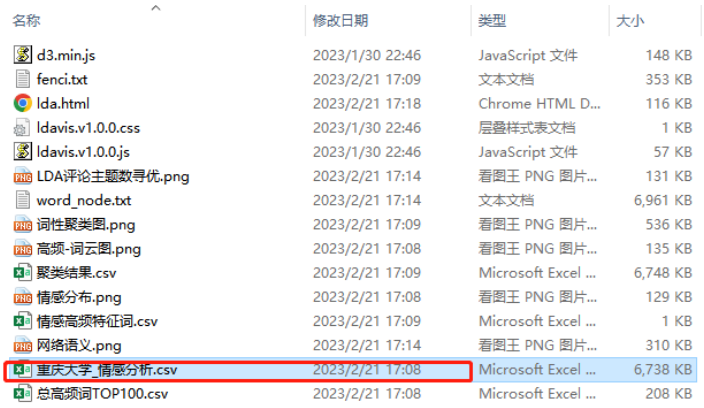

接着我们再去把TOP100的词用pandas保存下来，放在文件夹里面

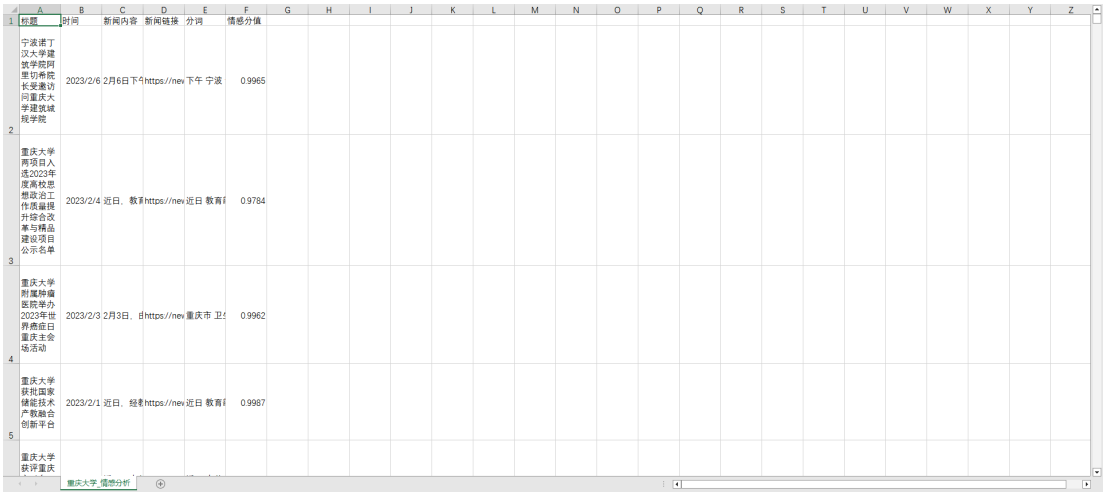



然后我们再去对文本进行情感分析，这里情感分析我们采用的是百度开源成熟的NLP模型库，paddlehub

对应的文档介绍：https://www.paddlepaddle.org.cn/hubdetail?name=senta_bilstm&en_category=SentimentAnalysis

通过这个库去对文本内容进行打分




这样我们就可以得知每个正面的具体情感分值是怎么样的一个情况，这里越接近1则是说明情感趋于积极，越接近0则说明情感趋于消极

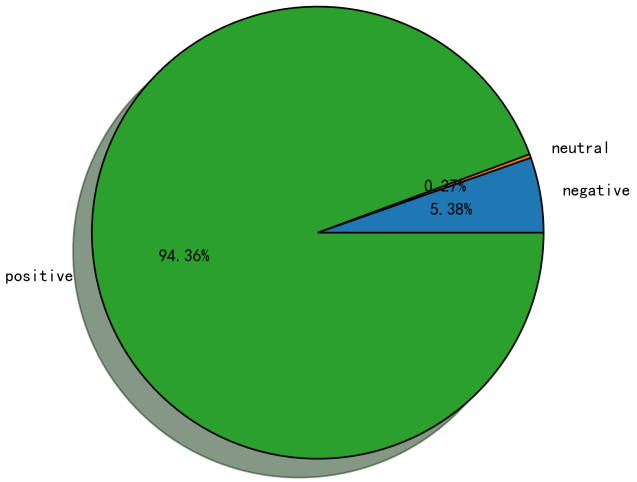
最后我们再用代码去划分积极，消极，中立，这里小于等于 0.45则是消极，在0.45-0.55区间则是中立，大于等于0.55则是积极

```
def main1(x):
    x1 = float(x)
    if x1 <= 0.45:
        return 'negative'
    elif 0.45 < x1 < 0.55:
        return 'neutral'
    else:
        return 'positive'
```

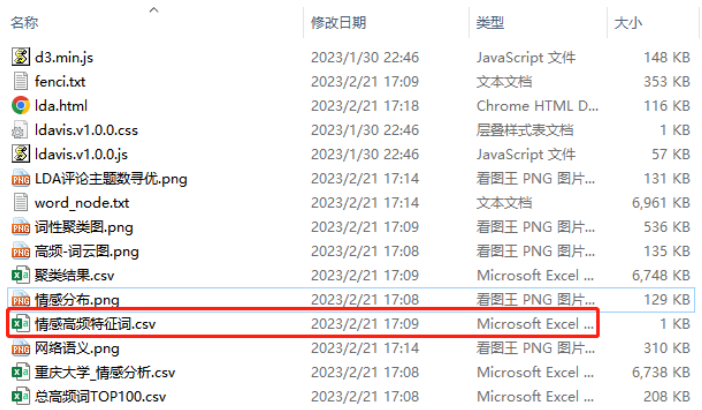
区分好之后，我们再用pandas中的value_counts()函数去统计，积极，消极，中立它们分别出现的频次

统计好之后，我们调用matplotlib中的pie函数去画饼图

最后的呈现效果，因为正面是官方发布的，所以大部分情感主要是趋于正面的居多，消极的和中立偏少，符合我们的常识，故此分类结果较为合适，这里为什么要用百度的开源情感模型而不是采用snownlp这个采用的情感分析库呢，主要还是正文这个内容，如果是调用snownlp这个情感分析模型，则几乎打分全是1，这样是没有任何参考价值的，准确率不太能继续分析下去，所以就调用了百度的开源情感模型

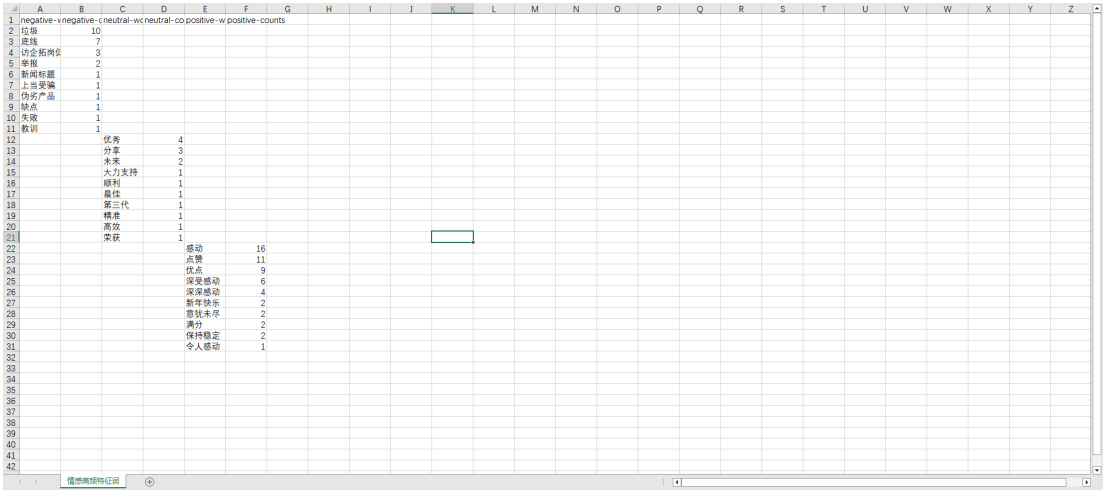



在得到对应的情感分析之后，我们则是去查看不同类型，它们对应出现的高频词情感分值最高的是什么词，这里情感分值，积极的则是按照情感分值越大，那这个词越靠前，消极的则是分值越小，那么这个词越靠前，中立的规则则是和积极的一样



| 名称 | 修改日期 | 类型 | 大小 |
| --- | --- | --- | --- |
| d3.min.js | 2023/1/30 22:46 | JavaScript 文件 | 148 KB |
| fenci.txt | 2023/2/21 17:09 | 文本文档 | 353 KB |
| lda.html | 2023/2/21 17:18 | Chrome HTML D... | 116 KB |
| ldavis.v1.0.0.css | 2023/1/30 22:46 | 层叠样式表文档 | 1 KB |
| ldavis.v1.0.0.js | 2023/1/30 22:46 | JavaScript 文件 | 57 KB |
| LDA评论主题数寻优.png | 2023/2/21 17:14 | 看图王 PNG 图片... | 131 KB |
| word_node.txt | 2023/2/21 17:14 | 文本文档 | 6,961 KB |
| 词性聚类图.png | 2023/2/21 17:09 | 看图王 PNG 图片... | 536 KB |
| 高频-词云图.png | 2023/2/21 17:08 | 看图王 PNG 图片... | 135 KB |
| 聚类结果.csv | 2023/2/21 17:09 | Microsoft Excel ... | 6,748 KB |
| 情感分布.png | 2023/2/21 17:08 | 看图王 PNG 图片... | 129 KB |
| 情感高频特征词.csv | 2023/2/21 17:09 | Microsoft Excel ... | 1 KB |
| 网络语义.png | 2023/2/21 17:14 | 看图王 PNG 图片... | 310 KB |
| 重庆大学_情感分析.csv | 2023/2/21 17:08 | Microsoft Excel ... | 6,738 KB |
| 总高频词TOP100.csv | 2023/2/21 17:08 | Microsoft Excel ... | 208 KB |

这里到时候需要你这边移动一下，因为代码的逻辑缘故，只能是按照这种方式呈现



| negative-v | negative-c | neutral-wo | neutral-co | positive-w | positive-counts |
| --- | --- | --- | --- | --- | --- |
| 垃圾 | 10 | | | | |
| 底线 | 7 | | | | |
| 访企拓岗促 | 3 | | | | |
| 举报 | 2 | | | | |
| 新闻标题 | 1 | | | | |
| 上当受骗 | 1 | | | | |
| 伪劣产品 | 1 | | | | |
| 缺点 | 1 | | | | |
| 失败 | 1 | | | | |
| 教训 | 1 | | | | |
| | | 优秀 | 4 | | |
| | | 分享 | 3 | | |
| | | 未来 | 2 | | |
| | | 大力支持 | 1 | | |
| | | 顺利 | 1 | | |
| | | 最佳 | 1 | | |
| | | 第三代 | 1 | | |
| | | 精准 | 1 | | |
| | | 高效 | 1 | | |
| | | 荣获 | 1 | | |
| | | | | 感动 | 16 |
| | | | | 点赞 | 11 |
| | | | | 优点 | 9 |
| | | | | 深受感动 | 6 |
| | | | | 深深感动 | 4 |
| | | | | 新年快乐 | 2 |
| | | | | 意犹未尽 | 2 |
| | | | | 满分 | 2 |
| | | | | 保持稳定 | 2 |
| | | | | 令人感动 | 1 |

情感高频特征词

# 接着对文本进行一个聚类处理

聚类参考文献： https://zhuanlan.zhihu.com/p/78798251

首先我们还是先分词，把无效词给除去掉，

然后我们先把文本中的词转化为矩阵，这里采用的是sklearn中的CountVectorizer

接着再去统计每个词中的tf-idf权重，这里采用的是sklearn中的TfidfTransformer

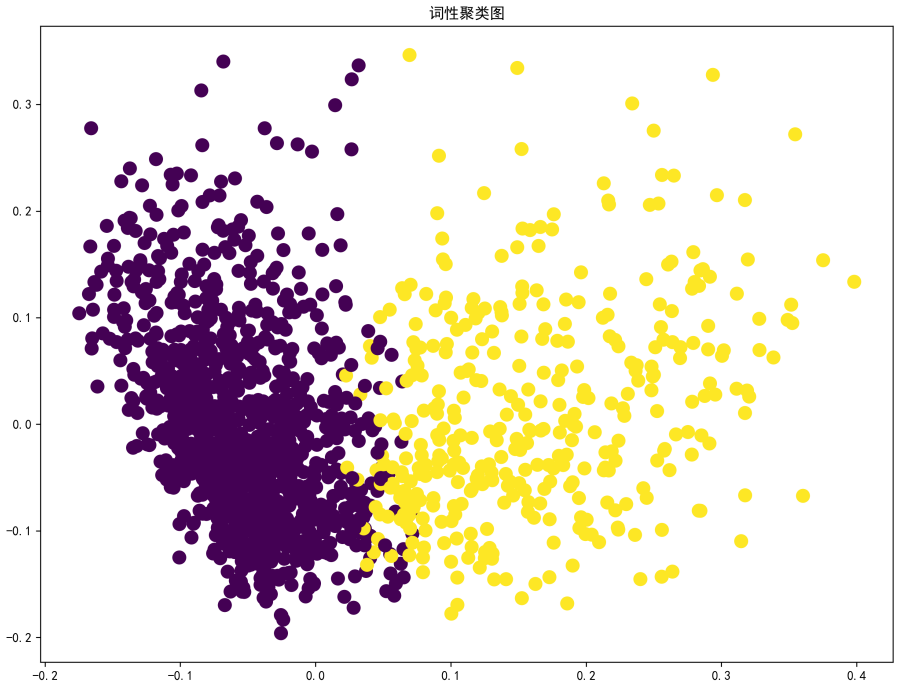
然后根据计算出来的权重和矩阵，进行平滑处理，也就是sklearn中的fit_transform

最后得出来的数据，我们再采用sklearn中的KMeans进行聚类

聚类结束后，我们先用PCA模块进行降维处理

接着再用matplotlib可视化出来，得出两个聚类后的结果展示图，这两个聚类，可以简单看着一个是非负，一个是负面，它们聚类的效果是怎么样的，这里只能清晰的判断，聚类的效果如何

并不能作为其他评判方式，这里之所以要采用聚类，是因为聚类是一种无监督的学习模型，在没有人为干预的情况下，可以得知文本的分类情况如何，像该图，基本上不会出现点重合的现象，那么这样一来的目的就是文本的分类情况较好，我们则可以根据这个结论去做下面的LDA主题建模，像聚类实际应用更多也是文本分类的应用，例如说我像快速的把一堆的文本分成几大类，然后我再根据这几个类去做一些划分啥的，那么聚类可以很好的帮助我们实现我们的这个目的，这样也不用人为的去打标签，再去训练，帮助我们节省一大堆的时间




# 接着我们再进行LDA主题建模

LDA参考文献： https://zhuanlan.zhihu.com/p/75222819

https://zhuanlan.zhihu.com/p/76636216

这里我们的步骤和上面和一样

首先我们还是先分词，把无效词给除去掉

接着我们开始构造主题数，寻找最优主题数，这里采用困惑度严格来说，判断标准并不合适，基于此我们这里采用的是另一种方式，也就是通过各个主题间的余弦相似度来衡量主题间的相似程度

**（2）寻找最优主题数**

基于相似度的自适应最优LDA模型选择方法，确定主题数并进行主题分析。实验证明该方法可以在不需要人工调试主题数目的情况下，用相对少的迭代，找到最优的主题结构。具体步骤如下。

① 取初始主题数k值，得到初始模型，计算各主题之间的相似度（平均余弦距离）。

② 增加或减少k值，重新训练模型，再次计算各主题之间的相似度。

③ 重复步骤②直到得到最优k值。

利用各主题间的余弦相似度来度量主题间的相似程度。从词频入手，计算它们的相似度，用词越相似，则内容越相近。
假定A和B是两个n维向量，A是，B是，则A与B的夹角θ的余弦值通过式（4）计算。

$$P（w_j \mid d_j）= \sum_{s=1}^{K} P（w_i|z=s） \times P(z=s|d_j)（3）$$

$$\cos\theta = \frac{\sum_{i=1}^{n} A_i B_i}{\sum_{i=1}^{n}(A_i)^2 \sum_{i=1}^{n}(B_i)^2} = \frac{AB}{|AB|}（4）$$

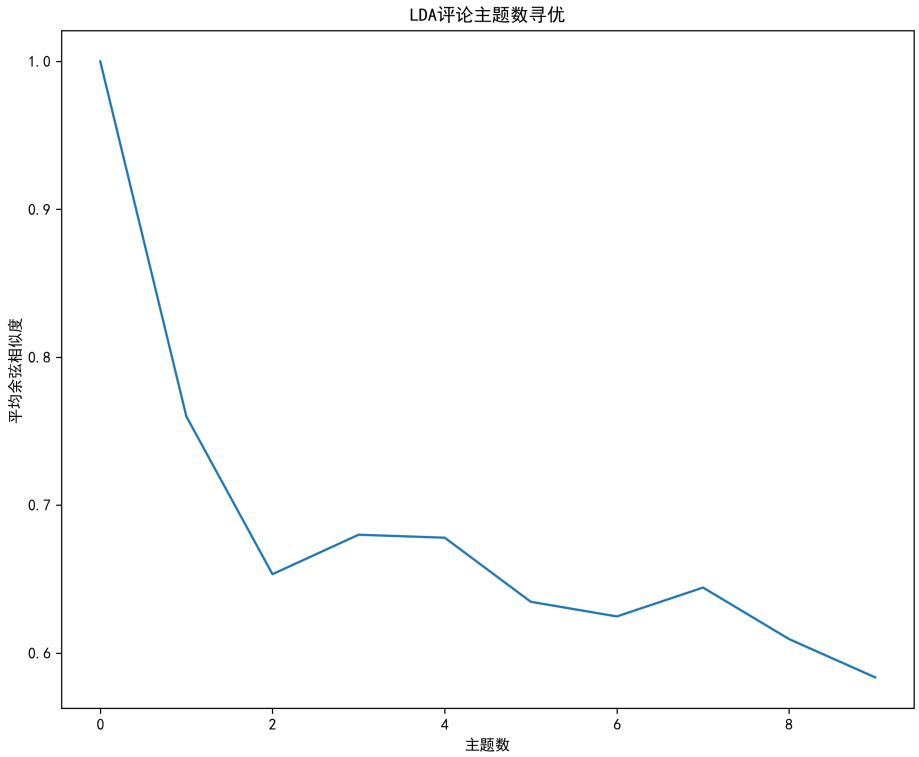
使用LDA主题模型，找出不同主题数下的主题词；每个模型各取出若干个主题词（比如前100个），合并成一个集合；生成任何两个主题间的词频向量；计算两个向量的余弦相似度，值越大就表示越相似；计算个主题数的平均余弦相似度，寻找最优主题数，如以下代码清单所示。

## 具体代码实现方式

```
# 构造主题数寻优函数
    def cos(vector1, vector2):  # 余弦相似度函数
        dot_product = 0.0
        normA = 0.0
        normB = 0.0
        for a, b in zip(vector1, vector2):
            dot_product += a * b
            normA += a ** 2
            normB += b ** 2
        if normA == 0.0 or normB == 0.0:
            return (None)
        else:
            return (dot_product / ((normA * normB) ** 0.5))

        # 主题数寻优

    def lda_k(x_corpus, x_dict):
        # 初始化平均余弦相似度
        mean_similarity = []
        mean_similarity.append(1)

        # 循环生成主题并计算主题间相似度
        for i in np.arange(2, 11):
            lda = models.LdaModel(x_corpus, num_topics=i,
id2word=x_dict)  # LDA模型训练
            for j in np.arange(i):
```

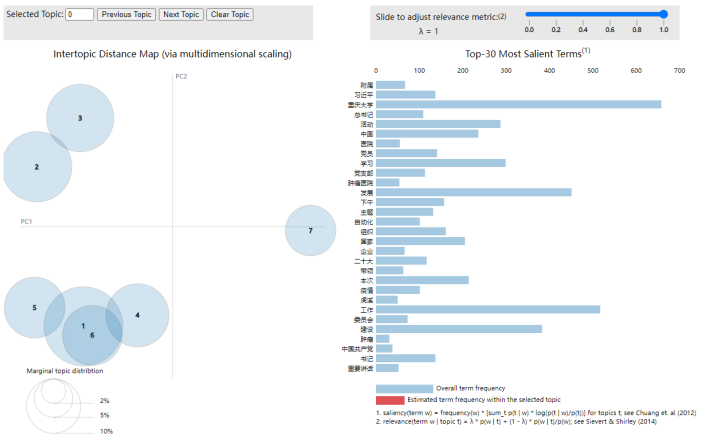
```
                term = lda.show_topics(num_words=30)

            # 提取各主题词
            top_word = []
            for k in np.arange(i):

                top_word.append(''.join(re.findall('"(.*)"', i)) \
                                for i in term[k][1].split('+')])  # 列
出所有词

            # 构造词频向量
            word = sum(top_word, [])  # 列出所有的词
            unique_word = set(word)  # 去除重复的词

            # 构造主题词列表，行表示主题号，列表示各主题词
            mat = []
            for j in np.arange(i):
                top_w = top_word[j]
                mat.append(tuple([top_w.count(k) for k in
unique_word]))

            p = list(itertools.permutations(list(np.arange(i)), 2))
            l = len(p)
            top_similarity = [0]
            for w in np.arange(l):
                vector1 = mat[p[w][0]]
                vector2 = mat[p[w][1]]
                top_similarity.append(cos(vector1, vector2))

            # 计算平均余弦相似度
            mean_similarity.append(sum(top_similarity) / l)
        return (mean_similarity)
```

处理好之后，再通过matplotlib来进行作图

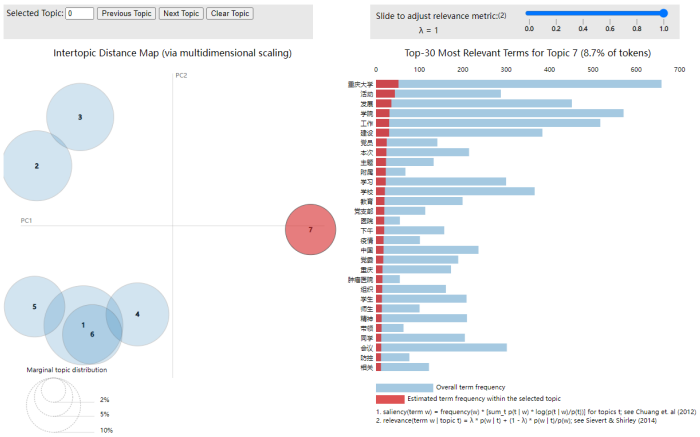



通过这种清晰的拐点，我们可以很好的判断对应的主题数是多少，这里可以看见从7之后，开始稳定下滑的趋势，那么其实4这里是一个转折点，我们可以根据这个拐点，把我们的主题数量定义为7

处理好之后，我们再采用gensim中的LdaModel去构建主题模型，最后把该主题用可视化的方式展示就是如图所示

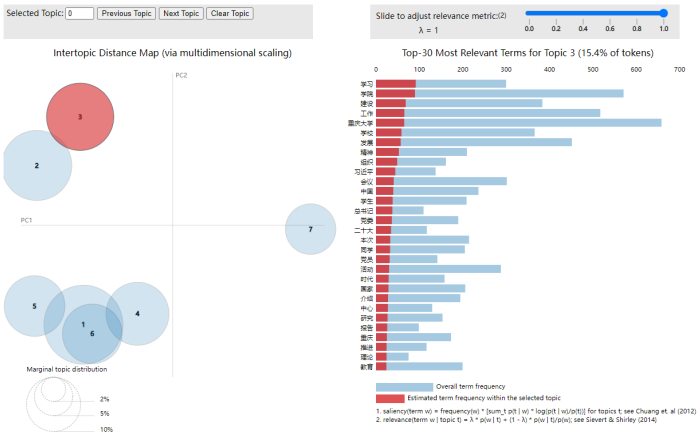


这里我们可以把鼠标放在对应的主题上，那么属于这个主题的词就变红，并且给出这些词的一个权重情况




每个主题的词的权重都不一样，也是通过这些词，我们可以进一步了解这个主题它的主题是什么，比如说像这里7说的则是医疗这个主题

像这里3的话，则是重庆大学关于二十大会议的相关主题内容




# 最后我们再来进行网络语义分析

背景：语义网络分析方法是以计算机为辅助的呈现和解释词语关系的文本分析方法。从术的角度来说，语义网络分析方法全面实现了传播学研究探求文本的表意、修辞及社会动因的研究目的，但从道的角度来看，其作为“元认知”的计算设计与传播学理论结合的前景尚不明朗。传播学研究在应用这一技术工具时，应意识到语义网络分析存在理论无关、忽视语法结构的缺陷，从而在具体的应用场景中对语义网络分析方法进行修正和补充，重视开发语义网络分析方法对于传播学研究的定制工具。只有术道相长，才能进一步提升语义网络分析方法对于传播学研究的理论价值。

应用：是用于数据挖掘的情感分析(sentiment analysis)。情感是人类传播的重要表达方式，侦测情感可以预测传播者的态度、行为倾向，因此被广泛应用于基于新闻文本的受众反馈研究、政治立场和意识形态研究等。语义网络分析方法通过有监督的注释可以完成对社交媒体文本的情感分类，其准确率接近人工编码的结果。

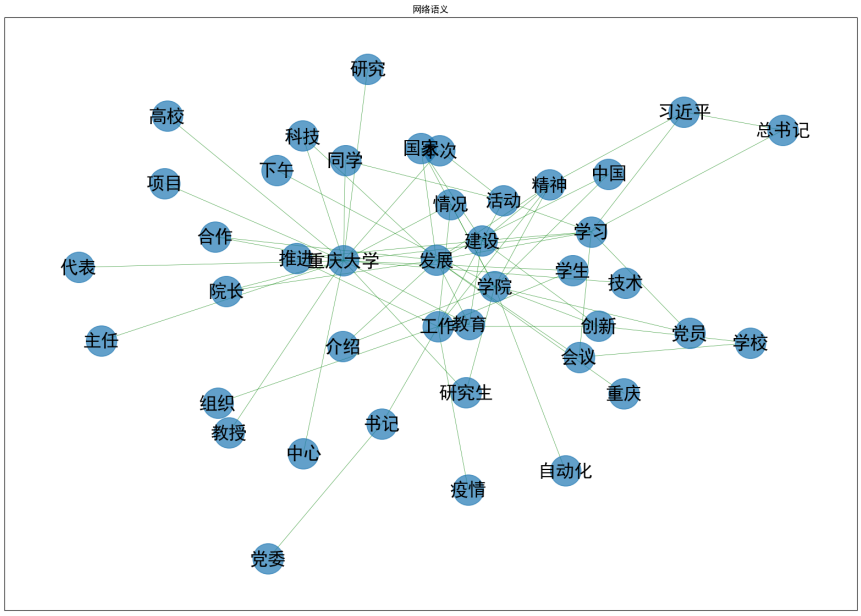
在这里我们先采用网络语义的基础算法构建

```
        while i < len(nums):  # ABCD共现 AB AC AD BC BD CD加1
            j = i + 1
            w1 = nums[i]  # 第一个单词
            while j < len(nums):
                w2 = nums[j]  # 第二个单词
                # 从word数组中找到单词对应的下标
                k = 0
                n1 = 0
                while k < len(word):
                    if w1 == word[k]:
                        n1 = k
                        break
                    k = k + 1
                # 寻找第二个关键字位置
                k = 0
                n2 = 0
                while k < len(word):
                    if w2 == word[k]:
                        n2 = k
                        break
                    k = k + 1

                # 重点: 词频矩阵赋值 只计算上三角
                if n1 <= n2:
                    word_vector[n1][n2] = word_vector[n1][n2] + 1
                else:
                    word_vector[n2][n1] = word_vector[n2][n1] + 1
                j = j + 1
            i = i + 1
```

构建好之后，我们再进行权重的筛选

把权重最高前100的词进行筛选出来，这里之所以只筛选前100是因为帮助我们更好的判断，哪些词的重要性，如果是全部都进行绘制，那么整张图是很乱，甚至是无法应用的，所以我们这里就筛选前100来查看

通过该图像，我们可以知道每个词它们之间的关联程度如何，哪个词是中心




**以上便是全部分析的思路和方法，这里每个大学的分析方法和思路都是一致的，只是文本内容以及部分代码需要进行修改，这里并不做过多的说明。**