**实验手册**

1. 前期准备

本次实验希望通过从数据清洗到分析和可视化的整个流程，让同学们直观地体验一个比较简单的文本分析项目，熟悉数据预处理的工作和分析可视化的三个工具，包括词云，热点地图和LDA主题分布。

本次实验在pycharm上运行，版本为PyCharm 2022.2.2 (Community Edition)，python的版本是3.10，并不需要额外配置其他环境。下面提供一个参考的版本信息，不必与其完全相同，可以正常使用的pycharm即可：

PyCharm 2022.2.2 (Community Edition)

Build #PC-222.4167.33, built on September 15, 2022

Runtime version: 17.0.4+7-b469.53 amd64

VM: OpenJDK 64-Bit Server VM by JetBrains s.r.o.

Windows 11 10.0

GC: G1 Young Generation, G1 Old Generation

Memory: 10776M

Cores: 16

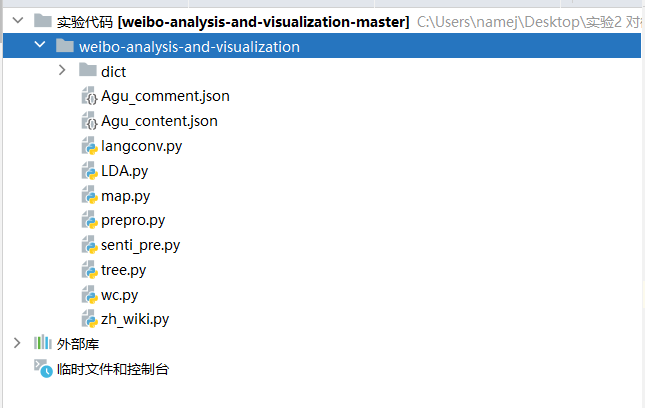
Non-Bundled Plugins:

com.tabnine.TabNine (1.8.0)

com.intellij.zh (222.426)

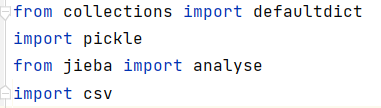
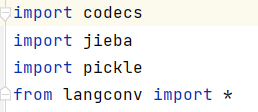
net.seesharpsoft.intellij.plugins.csv (3.1.0-222)

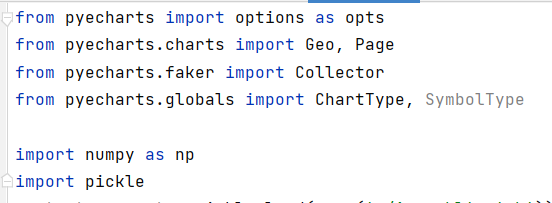
首先，我们在pycharm中打开项目，并将准备好的数据集（两个json文件，分别是微博中的内容以及评论）放入到weibo-analysis-and-visualization文件夹下。

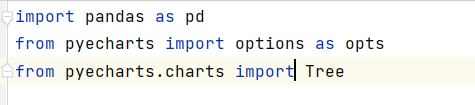
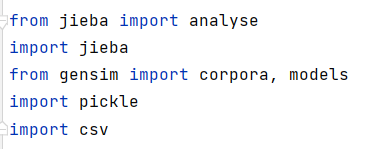


随后，导入实验所需要的库，如果电脑上没有的话可以在终端使用pip install <包名>，或者在pycharm中将鼠标悬停在报错信息上，点击解决方案中的导入xx包下载。

需要用到的安装包有（按照后续实验使用的文件顺序展示）：







编译器提示不再报错则可以进行后续的步骤。

1. 生成词云

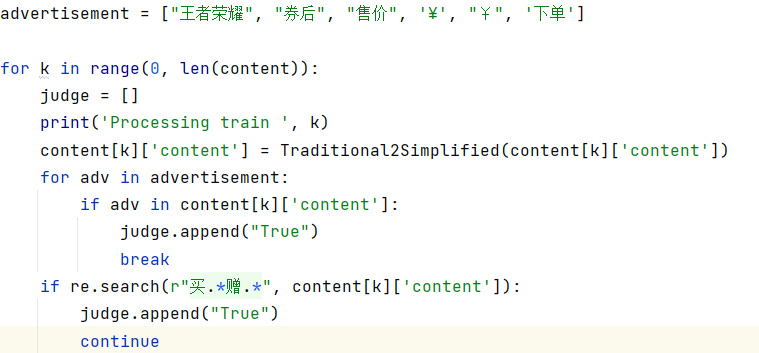
首先，运行prepro.py文件，进行数据预处理。从微博中抓取的内容和评论数据中包含一些乱码，敏感词和广告。prepro.py文件的功能是清洗数据，并且把内容和评论配对起来，方便之后生成词云和热度地图。

数据预处理包括了下面几个方面：

1.将繁体字转换为简体，包括转字和词，这部分工作是通过prepro.py中的Traditional2Simplified()函数调用langconv.py中的Converter类实现的，具体的类定义在langconv.py的第181-228行。



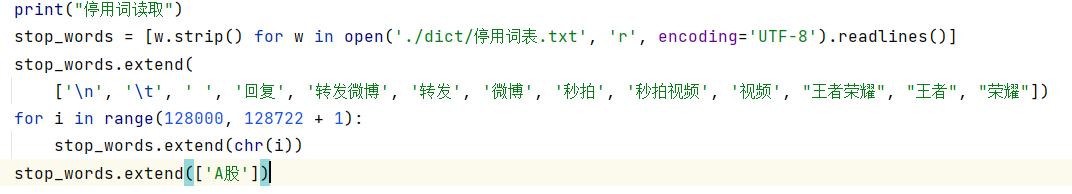
2.判断文本是否包含广告，并且剔除广告内容。主要通过一些广告中的词语对文本打上是否是广告的标签（judge）。



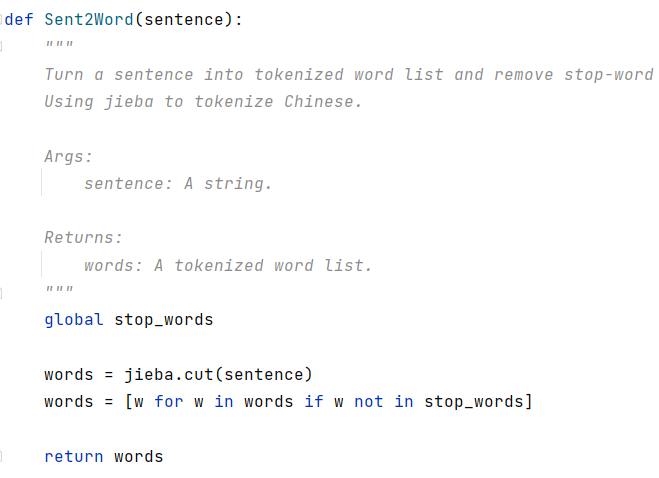
3.去除乱码。



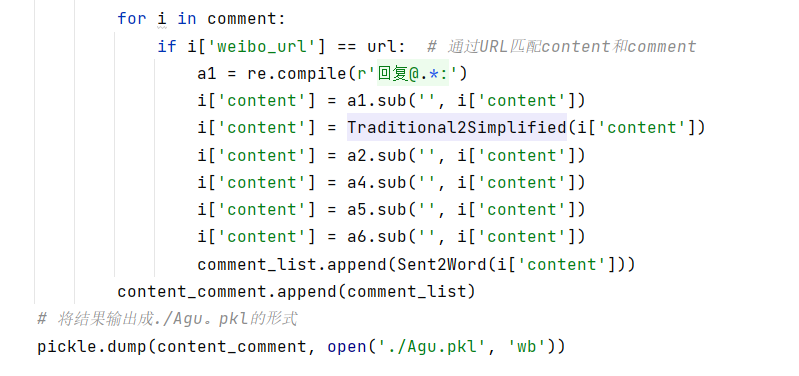
4.读取停用词，包括从./dict目录下读取的停用词文件以及手动根据情况添加的。



5.使用jieba进行分词，分词过程中去除停用词。



6.根据URL对微博的内容和评论进行拼合，输出成一个.pkl格式的文件。



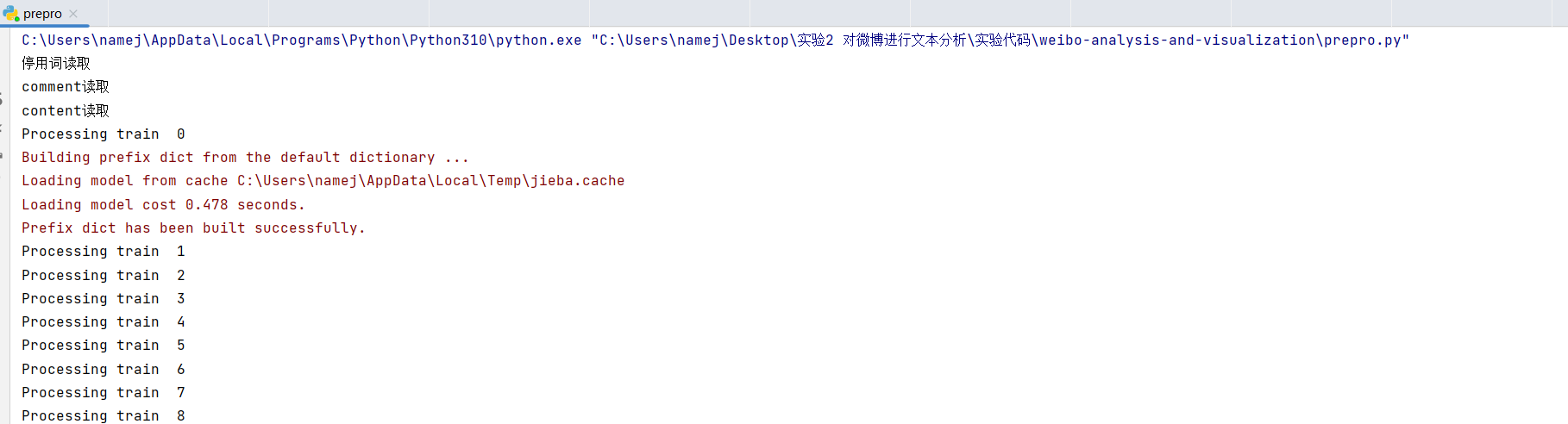
需要说明的是上面展示的预处理方法的顺序并不唯一，但按照合理的操作顺序处理，效果很可能更好。比如如果先分词再去除广告，那么广告句子中分出去的一些词语就不能被剔除，从而污染数据。

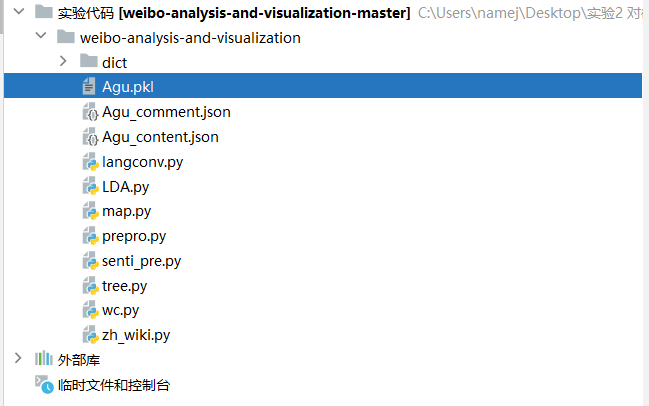
小结：在prepro.py文件中，我们将数据集进行了预处理，输出格式如下：

[  
 [url1, content1原文, [content1分词],[comment1],...,[comment\_n]],  
 [url2, content2原文, [content2分词],[comment1],...,[comment\_m]],  
 ...  
 ]

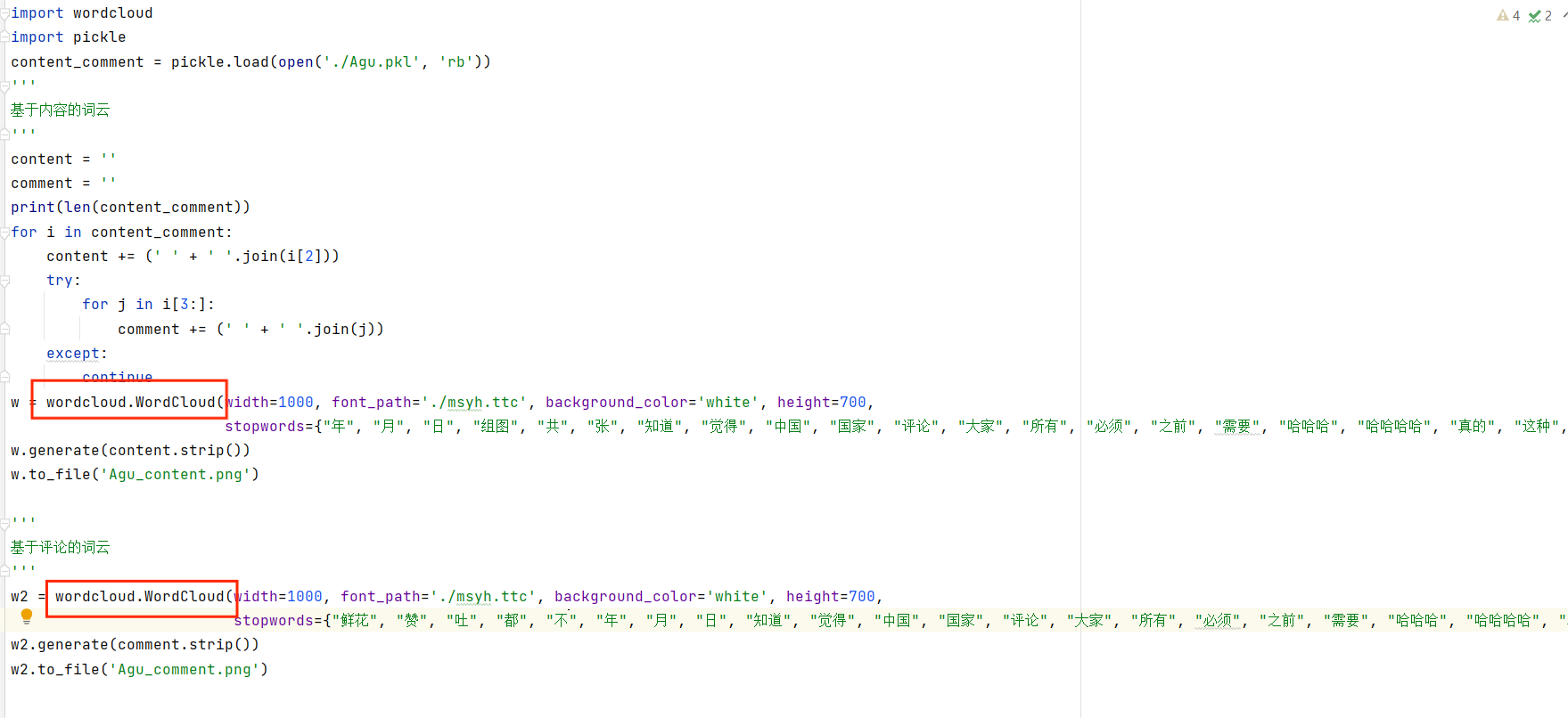
prepro.py运行成功后会发现，目录下多了一个Agu.pkl文件。

下面是运行成功的截图：





接下来使用wc.py进行词云的绘制。使用到的工具是wordcloud.WordCloud()函数，该函数的参数里包括了词云图的宽、高、字体、背景颜色等外观信息，以及比较重要的停用词，可以根据自己的需求来修改停用词。



运行成功后即可看到生成的词云图，内容和评论各有一张。

内容生成的词云

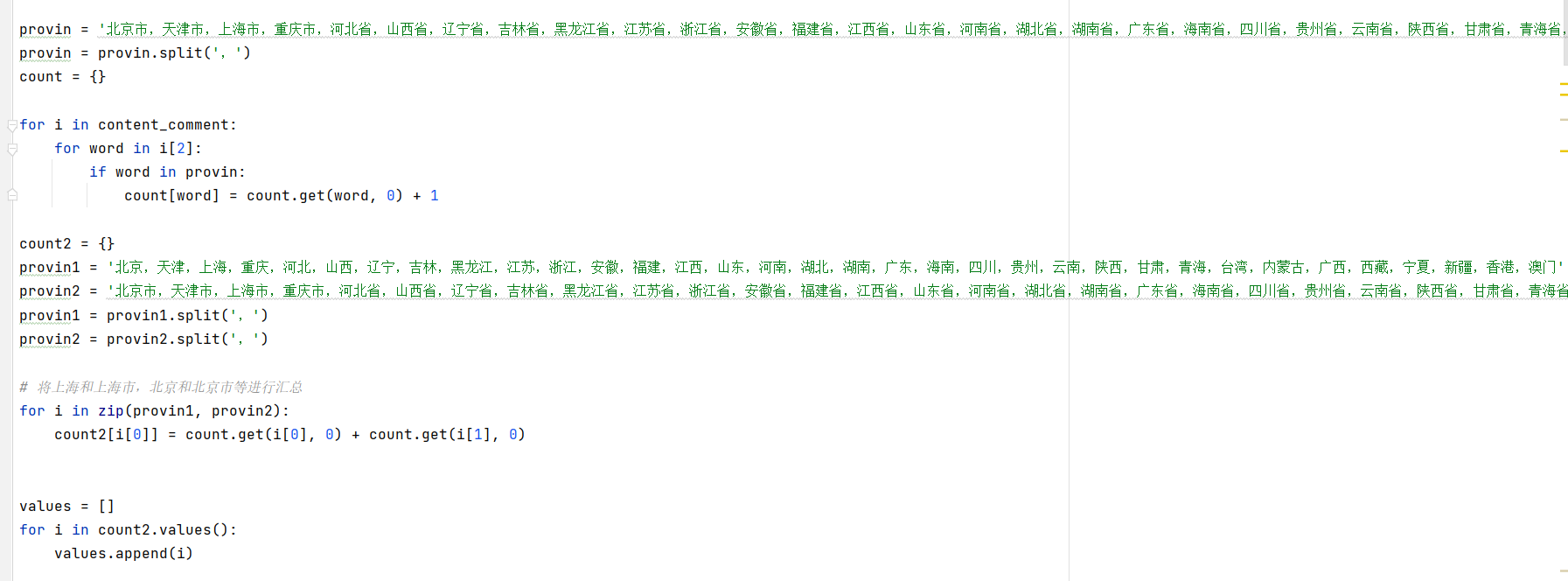
评论生成的词云

可以根据输出结果进一步修改停用词以迭代词云图。

1. 热度地图

热度地图使用的数据预处理与词云图部分相同，若目录中存在Agu.pkl文件，则可以直接运行map.py生成热度地图。

首先，合并城市和省份的信息。



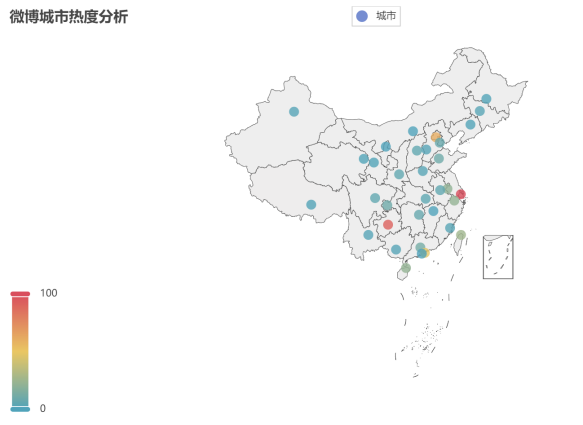
接下来，对城市的热度进行归一化处理，按最大热度100进行网比例调整。



随后使用，pyecharts中的Geo工具进行绘图。



热度地图生成成功后，目录下会多出一个html文件，打开就能看到结果：



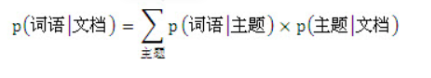
1. LDA主题分布

上述使用的两个文本分析工具（词云和热度地图）都很简单基础，下面我们引入一个更复杂的工具——LDA主题分布，对文本进行进一步分析。

LDA主题模型主要用于推测文档的主题分布，可以将文档集中每篇文档的主题以概率分布的形式给出根据主题进行主题聚类或文本分类。LDA主题模型不关心文档中单词的顺序，通常使用词袋特征（bag-of-word feature）来代表文档。

首先运行senti\_pre.py文件，进行数据预处理，这个预处理文件的方法与prepro.py大概相同，也包括了繁体转简体、判断并去除广告、去除乱码、读停用词、分词。区别在于此处只处理了内容content。

此处未处理的评论的原因是评论的文本长度很短，不同用户的评论之间逻辑性很弱，使用LDA主题分布的效果并不好。在LDA中，如果我们要生成一篇文档，它里面的每个词语出现的概率为：



看到文章推断其隐藏的主题分布，就是建模的目的。换言之，人类根据文档生成模型写成了各类文章，然后丢给了计算机，相当于计算机看到的是一篇篇已经写好的文章。现在计算机需要根据一篇篇文章中看到的一系列词归纳出当篇文章的主题，进而得出各个主题各自不同的出现概率：主题分布。

LDA的目的是对一篇文章归纳主题，而评论并不是一篇严格意义的文章，它可以看作很多很短的文章的集合，所以并不适合处理。

下面是senti\_pre.py中与prepro.py主要不同的代码（拼合输出部分）：



运行成功后目录中会多一个content2.pkl文件。

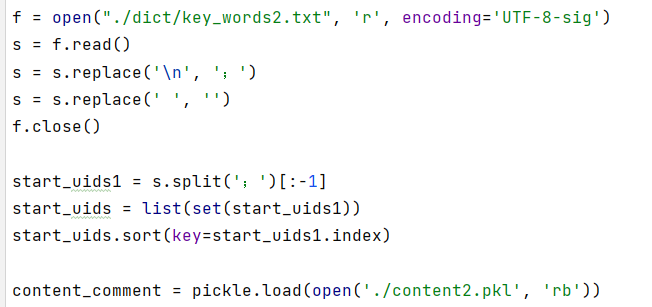
接下来，运行LDA.py进行LDA主题分布，这部分代码是实验重点，需要理解各个部分对应的功能：

1.添加自定义词汇，可以根据需要自行设置：

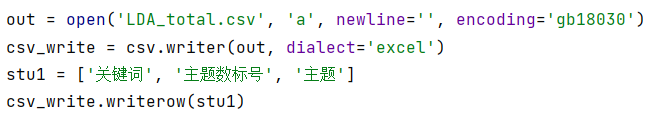


2.读取关键词文件和内容数据。

这部分将关键词数据处理为了一个包含关键词的列表start\_uids。



3.打开CSV文件以写入主题分析结果

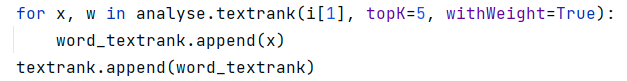


4.遍历内容数据，对每个关键词的评论进行处理，这部分代码通过一个外层循环遍历关键词列表start\_uids，然后内层循环在内容数据中查找匹配该关键词的内容，并对这些内容进行处理。



4.1.使用Textrank算法提取每条内容的关键词（前5个，topK参数指定）。

在内层循环中，使用Textrank算法提取每条内容i[1]中的关键词，限定提取前5个关键词，并将这些关键词存储在word\_textrank列表中。



4.2.创建一个字典（dictionary），将关键词映射为数字编号。这行代码创建了一个字典（dictionary）对象，用于将关键词映射为数字编号。textrank是包含关键词列表的列表，这将用于构建文档-词袋矩阵。



4.3.创建文档-词袋（document-term）矩阵，以便用于LDA主题建模。

在这里，通过dictionary.doc2bow方法，将关键词列表textrank转换为文档-词袋矩阵corpus。该矩阵表示了每个文档中的关键词及其频次，用于后续的LDA主题建模。

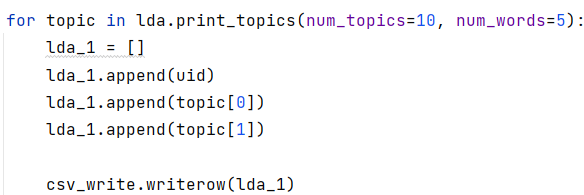


4.4.使用LDA模型对文档-词袋矩阵进行主题建模，生成主题分布。

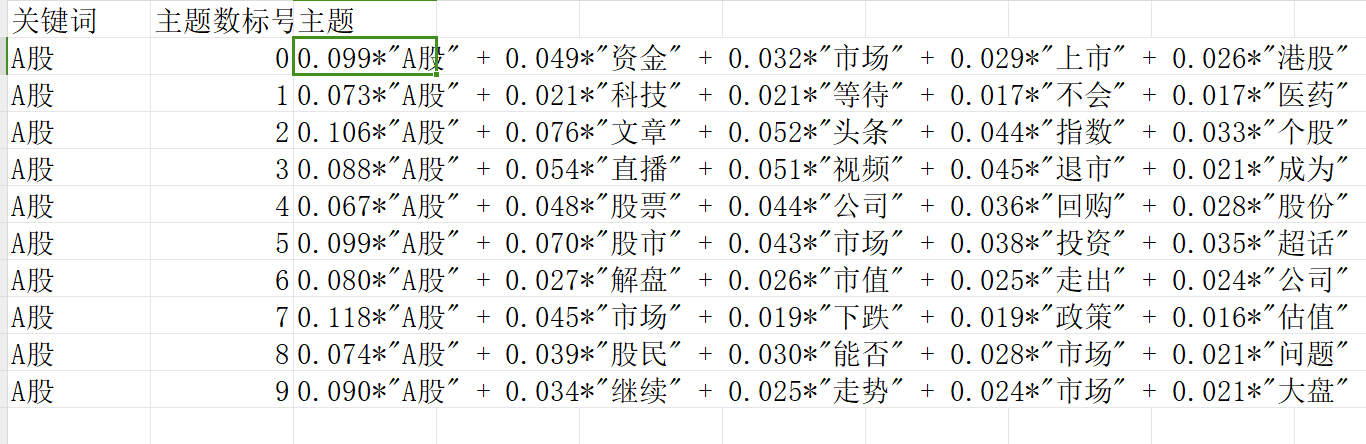
这行代码使用gensim库中的LDA模型对文档-词袋矩阵进行主题建模。corpus是文档-词袋矩阵，dictionary是词汇映射字典，num\_topics指定了主题的数量。LDA模型会生成主题分布，其中包含每个主题的词汇分布。



4.5.将主题分析结果写入CSV文件中。



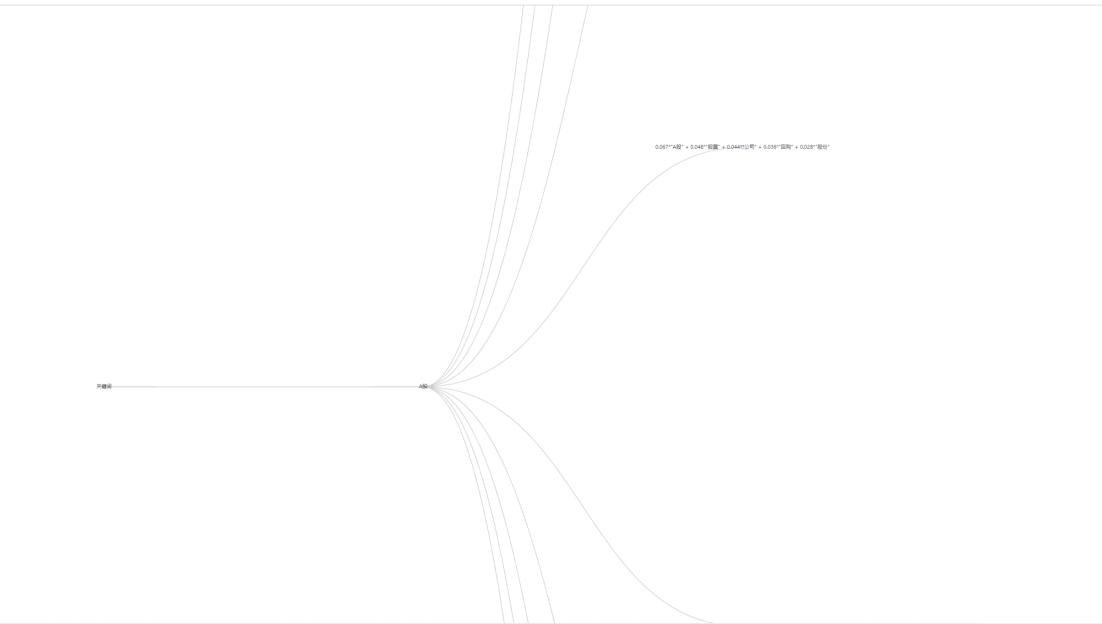
代码运行成功后，目录下会多出一个LDA\_total.csv，打开可以看到如下结果。



最后，运用tree.py对LDA的结果进行可视化，画出一个LDA主题树。主要使用的工具是pyecharts中的Tree工具。



运行完成后目录下生成一个html文件，打开后看到如下结果即为成功。一开始树没有展开，需要点一下A股那里的节点才能展开（图比较长，这里截取了一部分）：



至此，本次实验结束。