3022207128-杨宇鑫-实验报告4-2

1. 实验目的:

绘制可视化论文引用网络

节点为可视化领域的论文，链接为论文之间的引用。数据在vis\_paper\_citation\_network文件夹中。

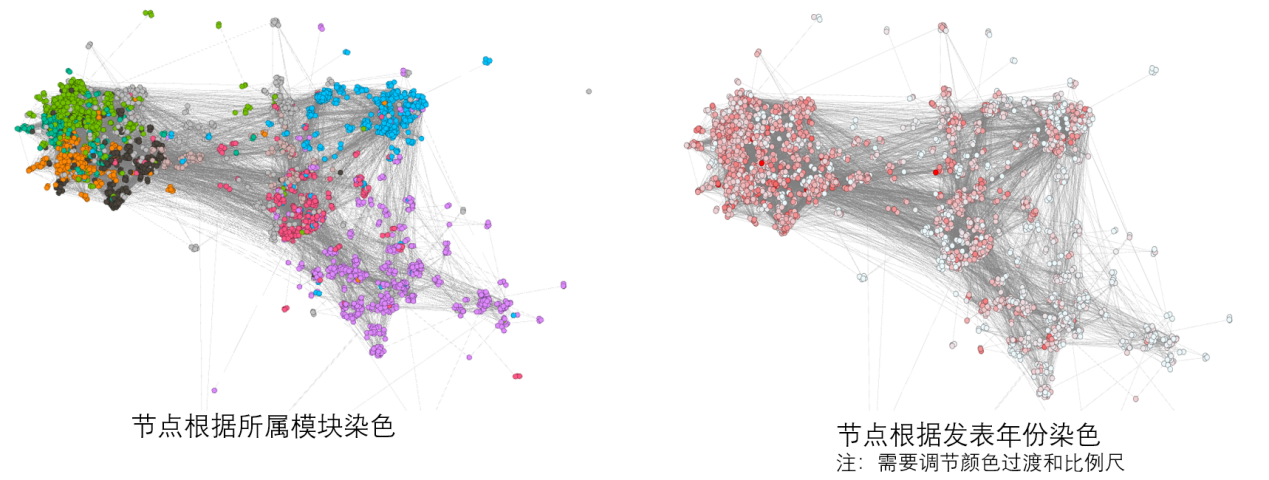
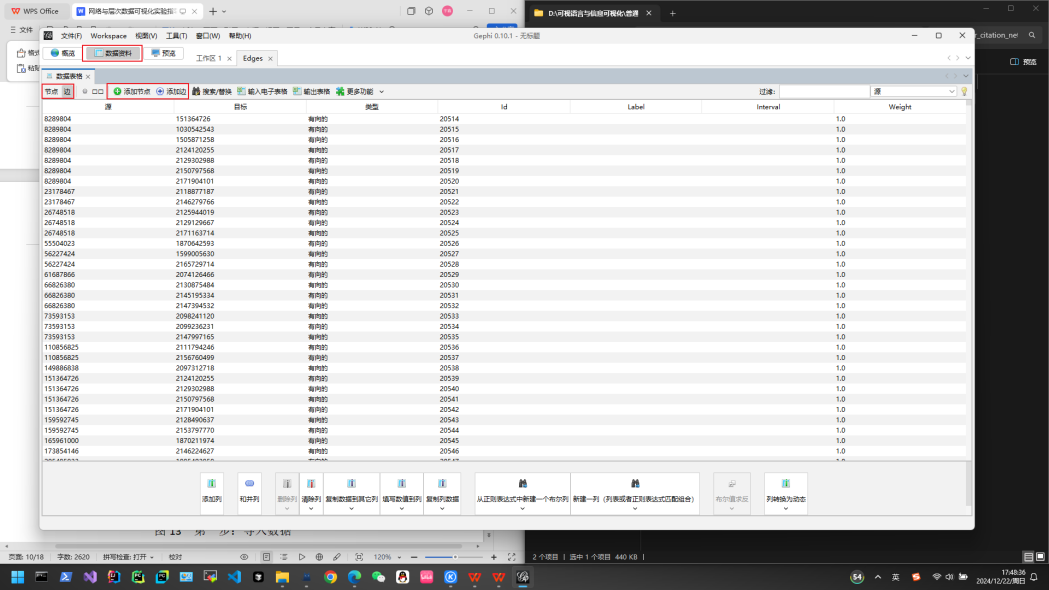
要求：使用Gephi绘制，效果参考下图 。请用Gephi的社区检测算法识别类别，导出识别结果。然后对node的Fields\_of\_Study字段进行词频统计，得出每个类别的语义。结合语义和图谱中节点的分布写出你的发现（不少于200字）。

图23 实验2效果参考图（此图布局算法使用的是OpenOrd，其他也可，只要能揭示出聚类结构）

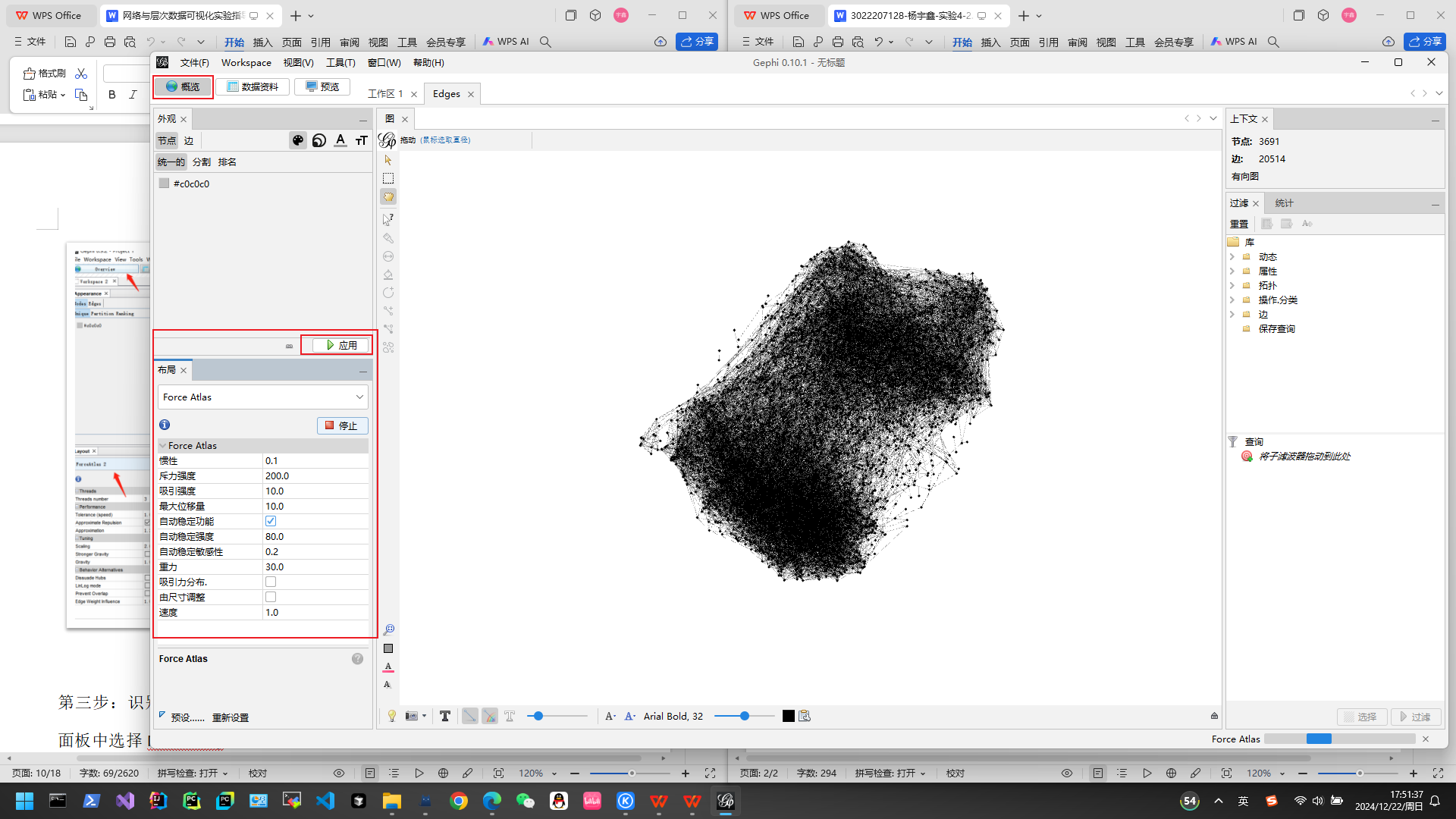
1. 实验过程:
   1. 下载Gephi网络可视化客户端软件

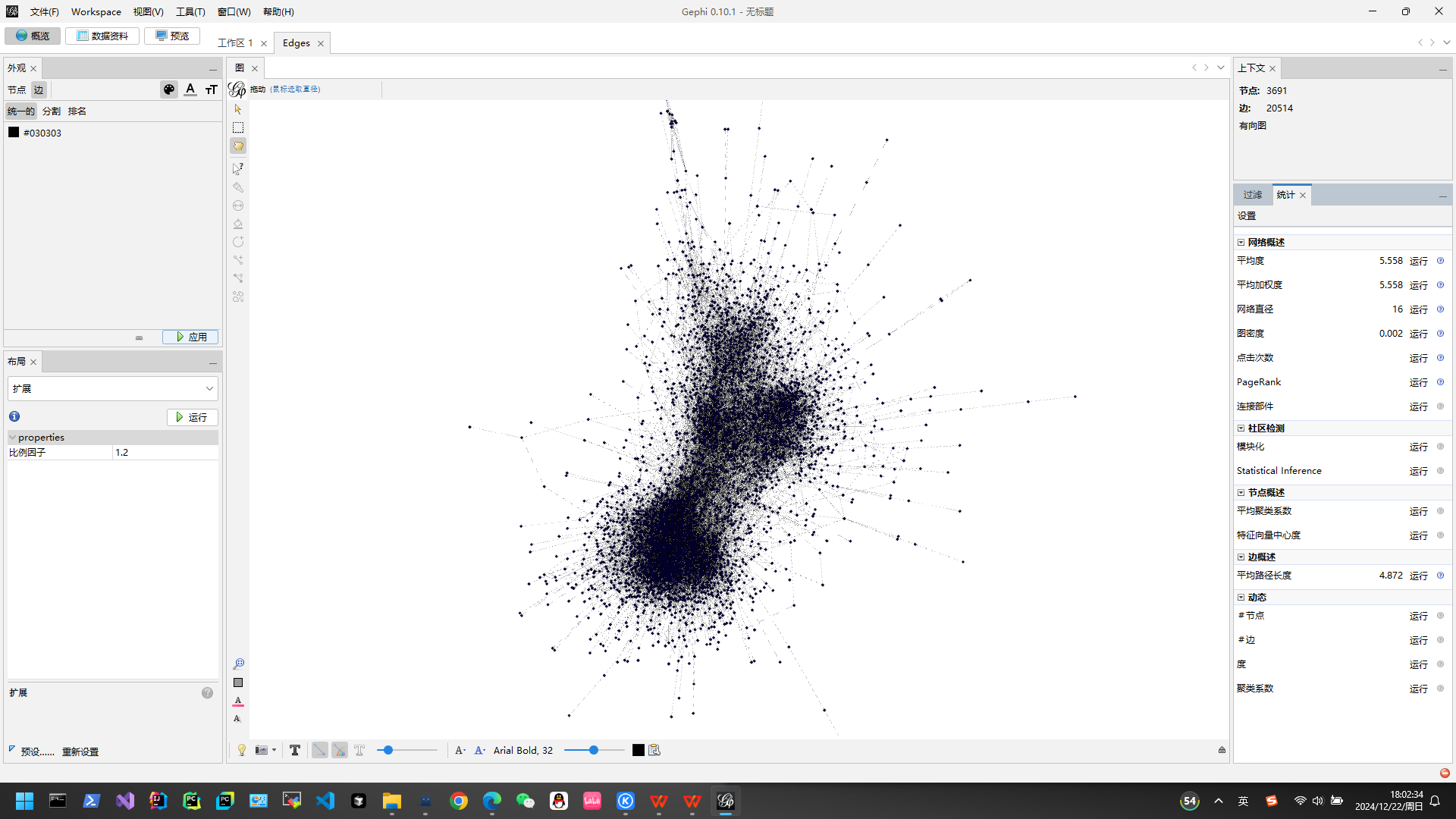


* 1. 文件->新建项目
  2. 在Data Laboratory模块中依次导入nodes.csv和edges.csv

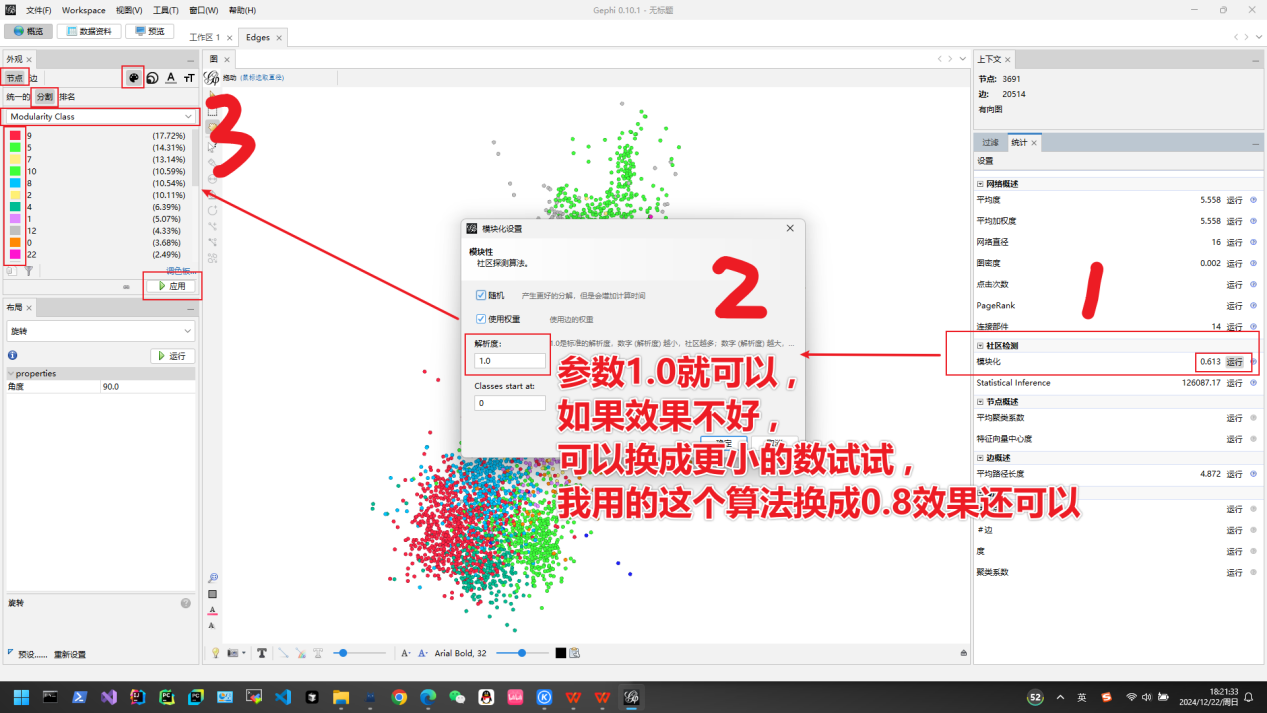


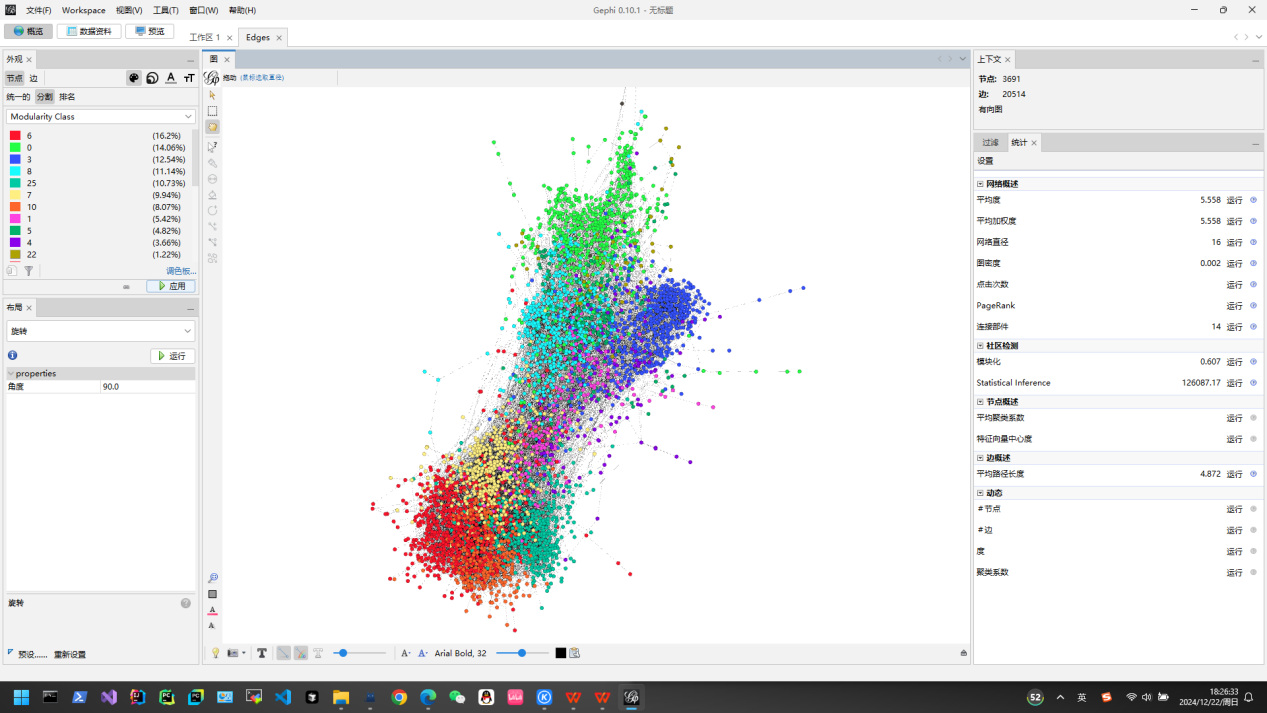
* 1. 布局。在Overview模块左侧的Layout面板中选择合适的布局算法，点击run会得到一个初始布局。若对布局结果不满意，则调整布局算法的参数或者选择其他的布局算法。





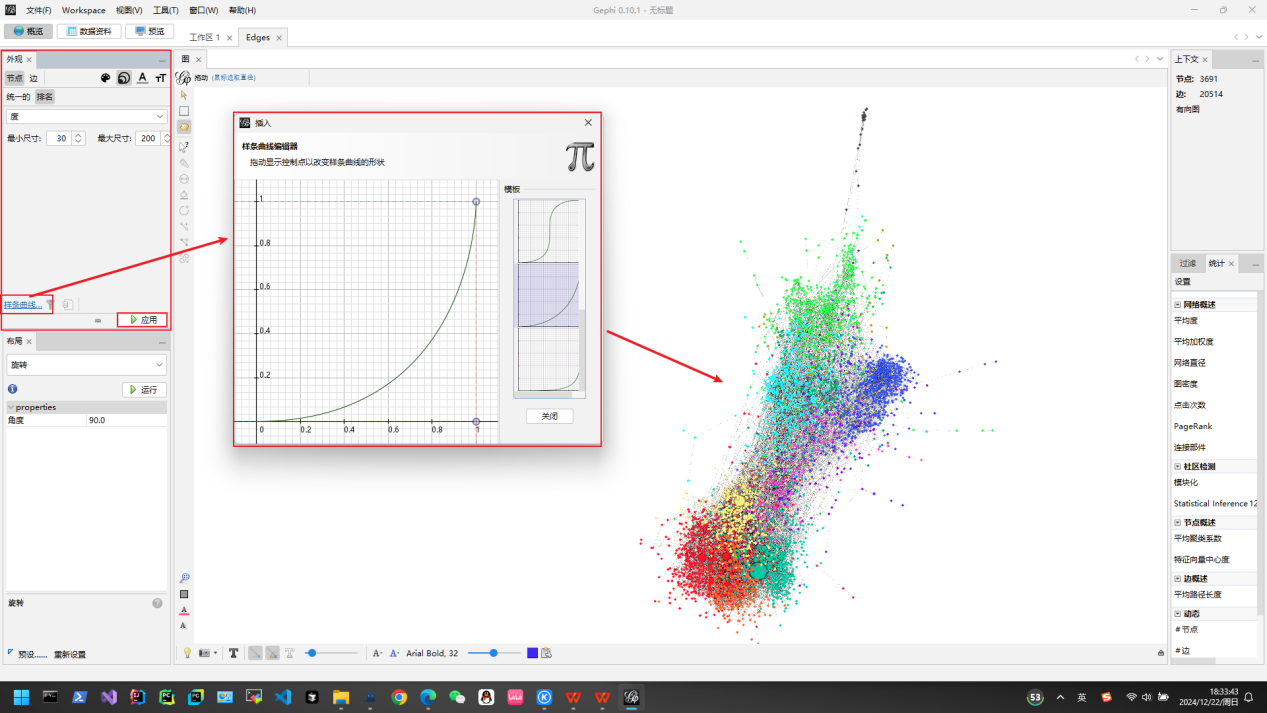
* 1. 识别群组并对节点染色。在Overview模块右侧的statistics面板中选择Modularity，点击run执行社区检测算法，根据检测结果调整算法参数，最好使得少于20个社区包含60%以上的数据点。在Overview模块左上角根据节点所属社区为其染色。

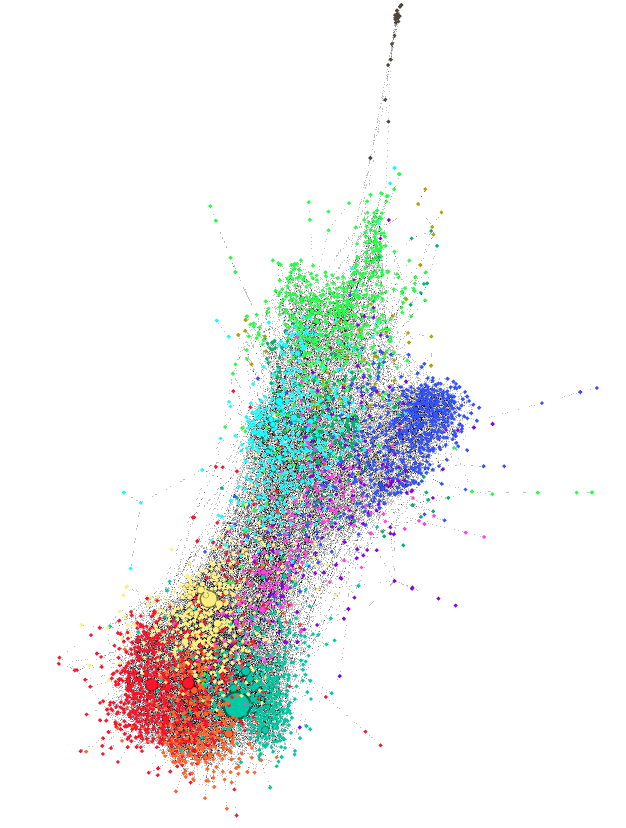




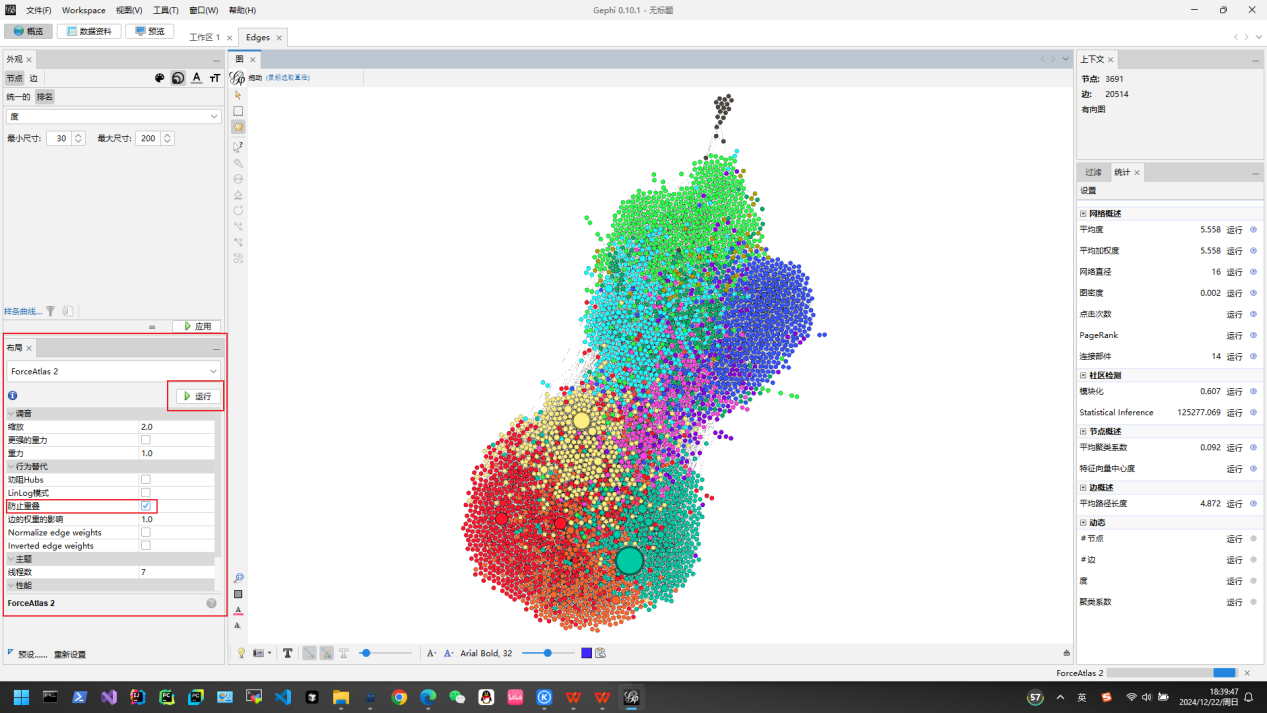
2.6 配置节点半径。在Overview模块左上角为节点设置不同的半径，比如度较大的节点半径较大。除了设定最大半径和最小半径之外，还可以点击spline设置所选度量到半径的映射(默认是线性映射)，以避免出现某些节点过大或者过小的情况。

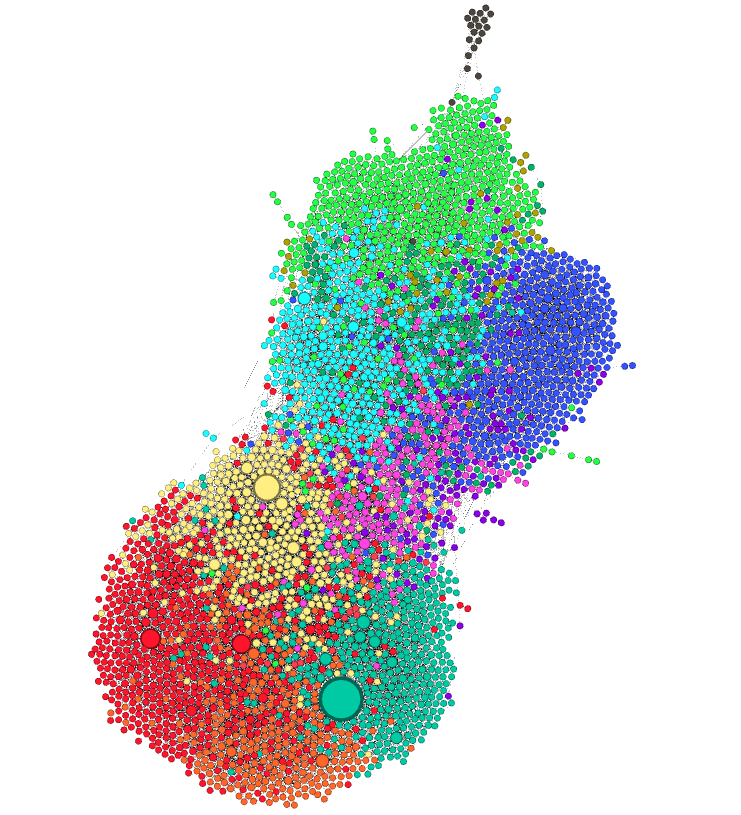
效果:





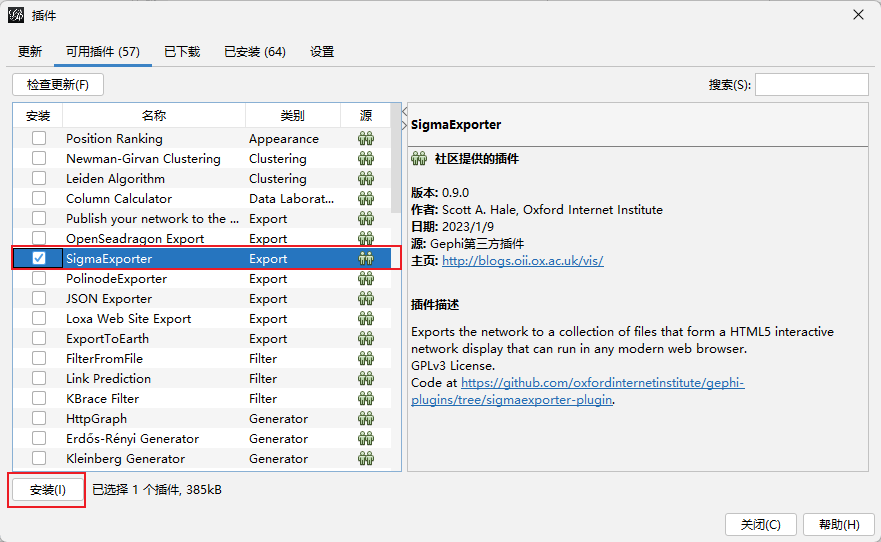
2.7 消除节点重叠。此时节点出现重叠，则回到布局算法界面，勾选Noverlap算法（似乎无效）或者带有prevent overlap选项的算法（勾选上此选项），再次点击run运行算法。节点位置较为稳定后，可点击stop提前终止算法。



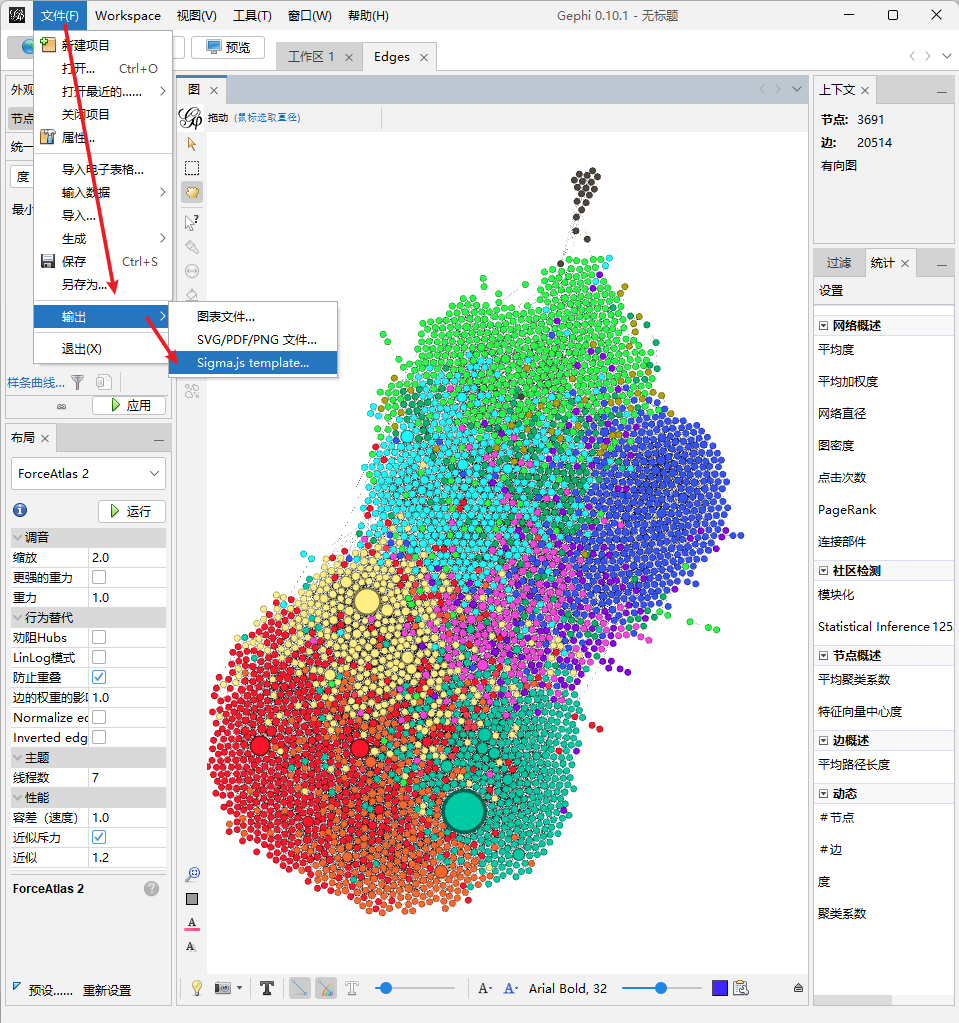


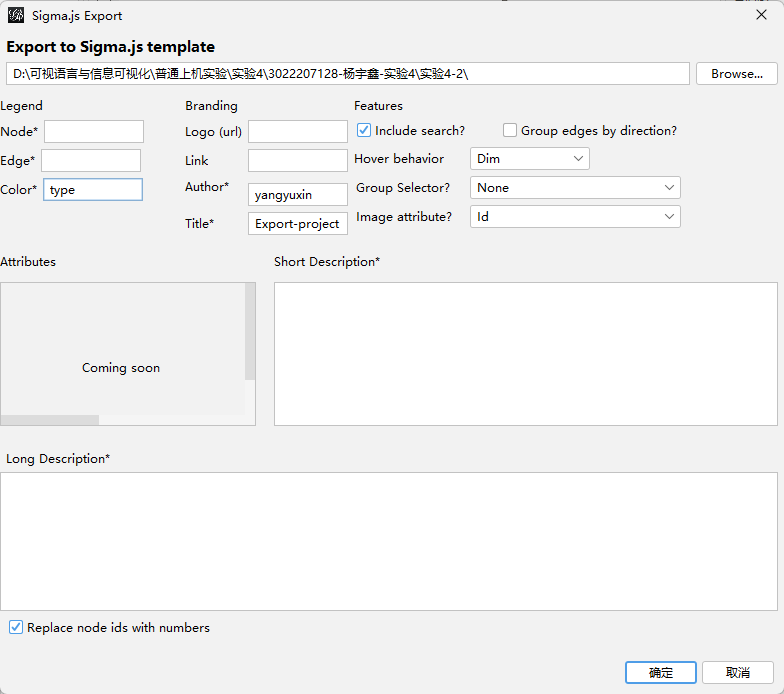
2.8 导出。可以在Preview中导出成png、svg或者pdf的静态格式，也可使用sigma.js插件导出一个基于web的交互式系统。此处选择后者。





重启软件 , 并将刚刚做出的模型保存一下 , 重启后选择刚才的项目





2.9 运行导出的结果。

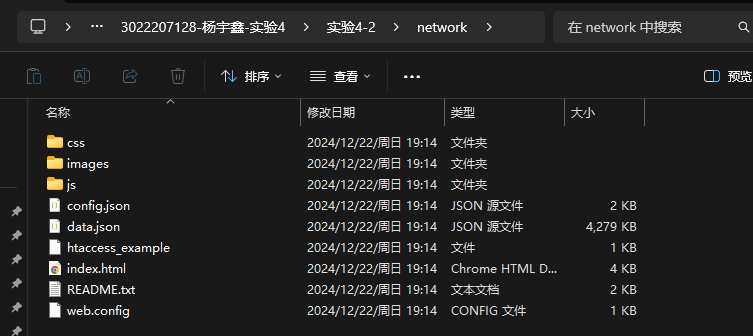
开启服务器：

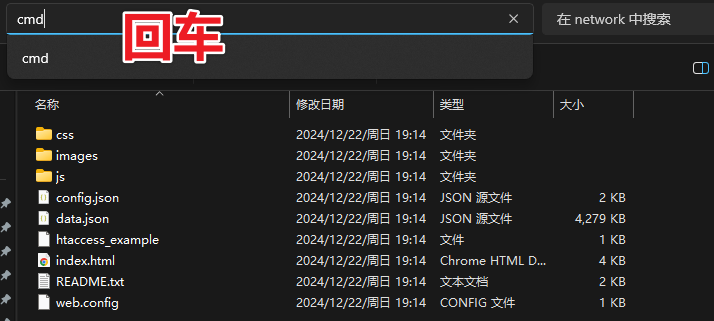
打开终端，切换到导出的文件夹目录

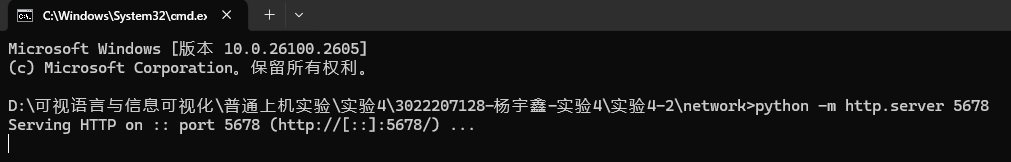
使用python3运行python –m http.server xxxx, xxxx为端口号，通常4500到9000之间任意数都可，注意如果同时运行两个程序需要使用不用端口号

在浏览器中打开<http://localhost:xxxx/index.html>， xxxx为上一步设置的端口号

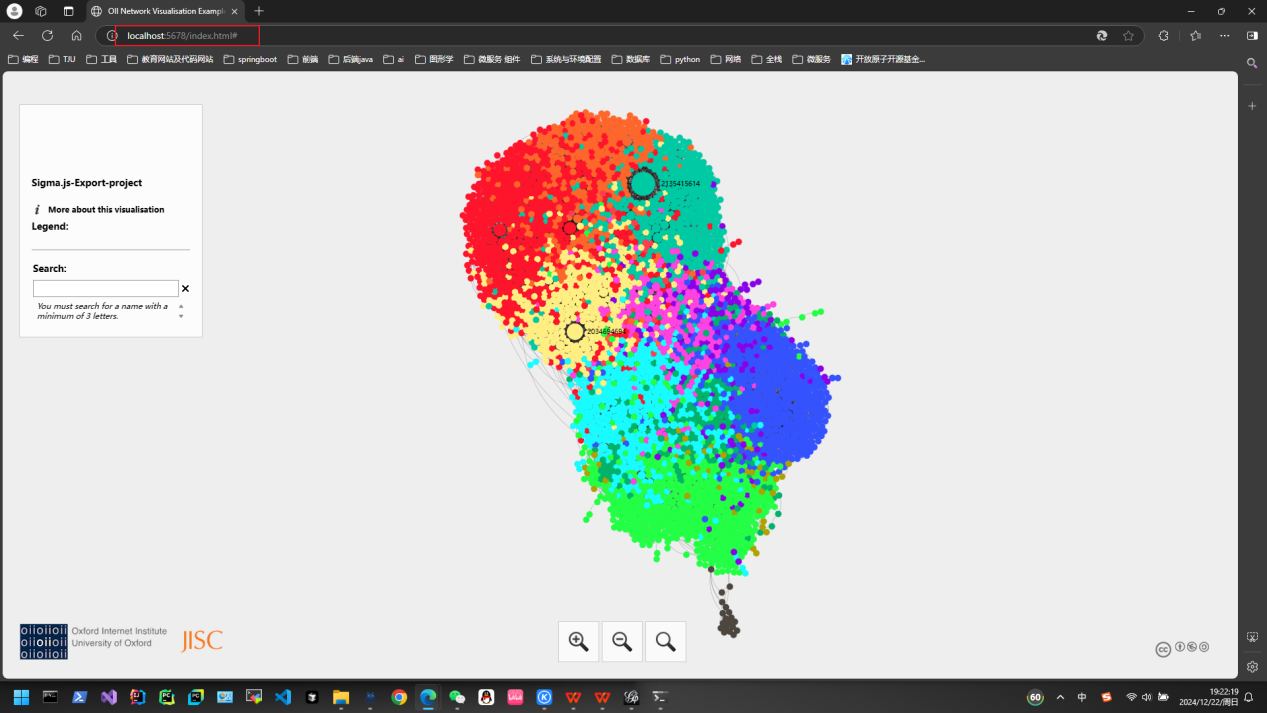
自此得到一个能在浏览器中良好运行的并带有丰富交互的网络可视化应用程序。交互包括：平移缩放、查找节点、点击节点查看其邻域等。







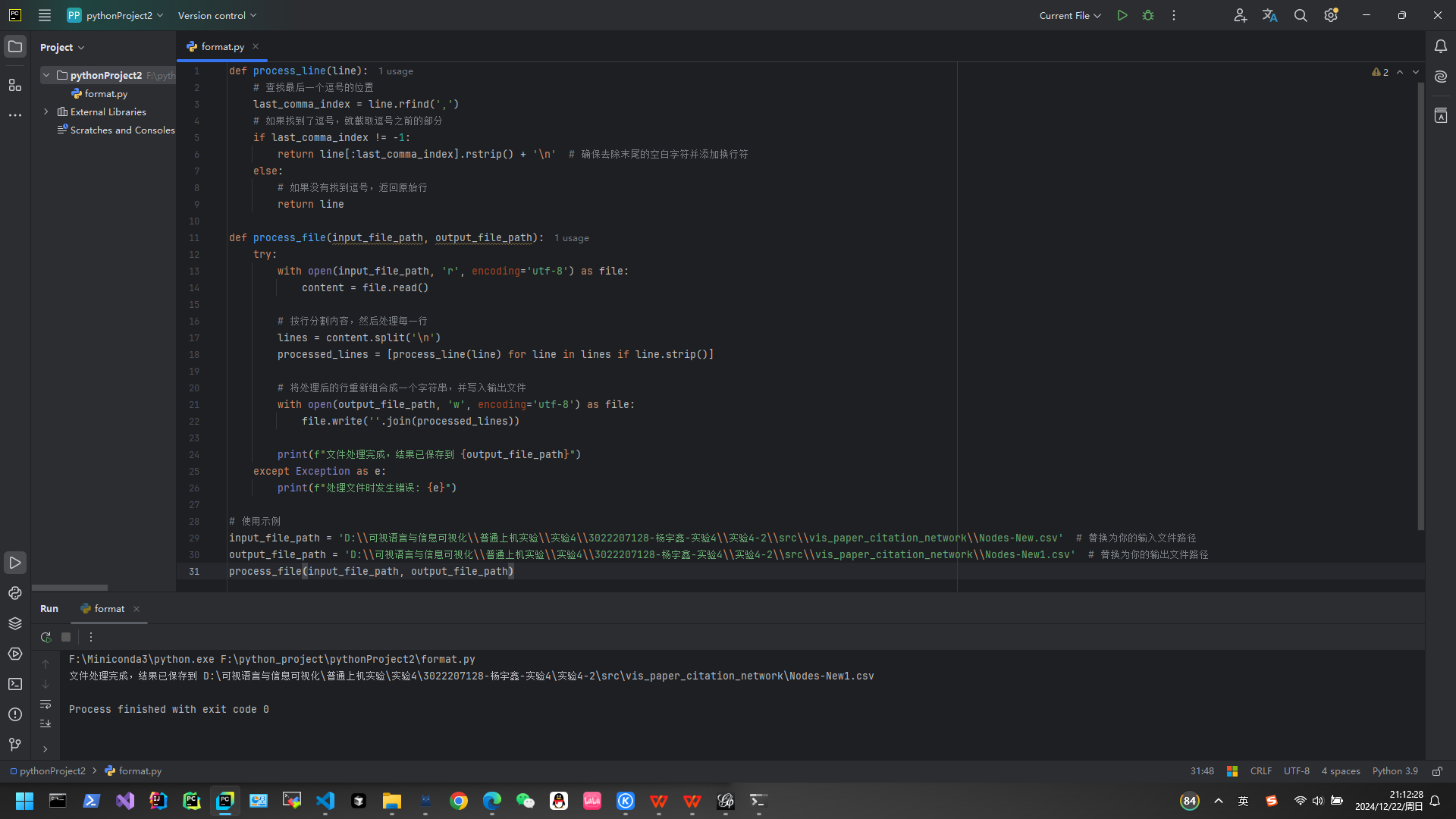
浏览器访问:



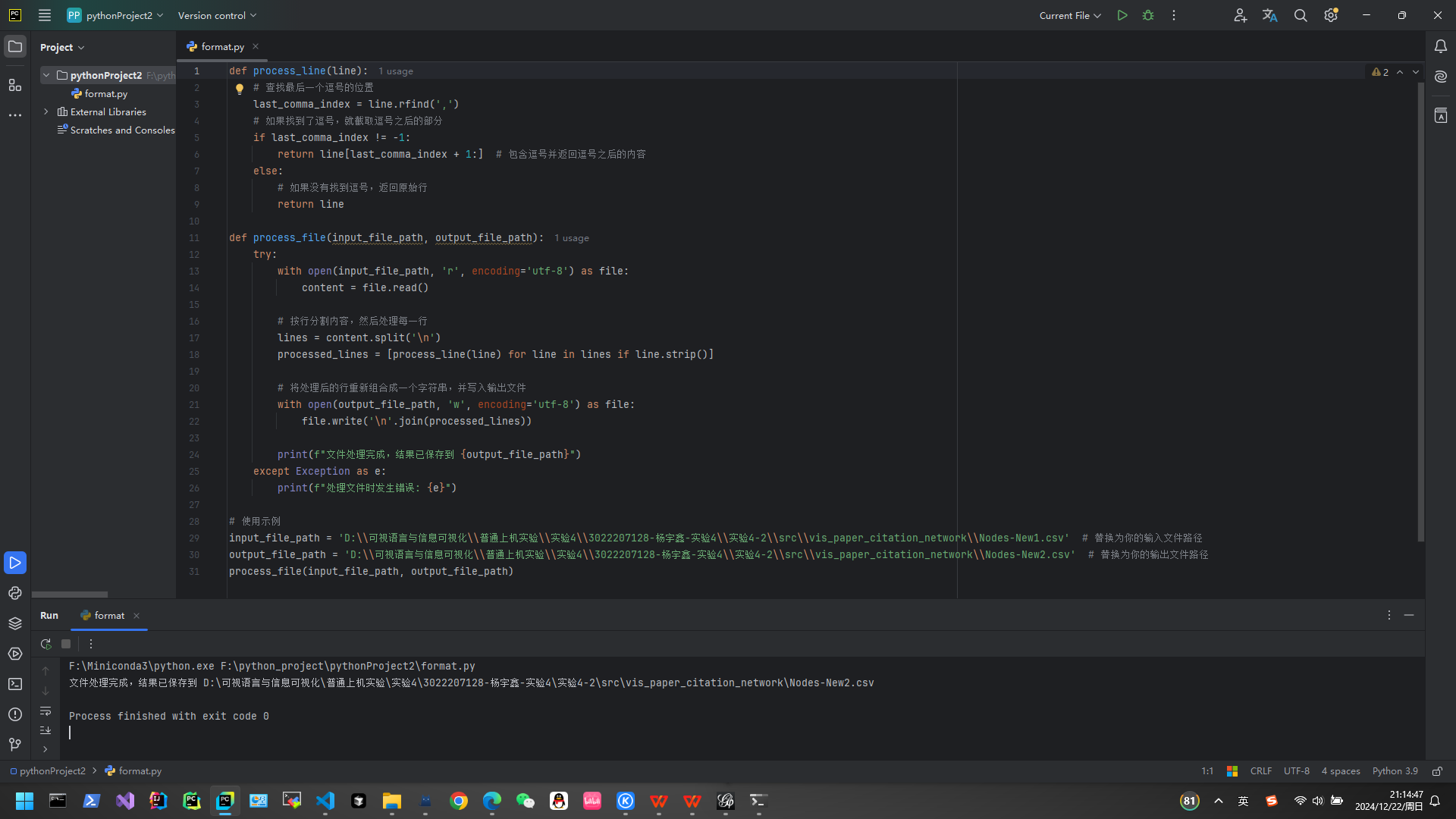
2.10 对node的Fields\_of\_Study字段进行词频统计

2.10.1 使用python脚本处理文件数据

def process\_line(line):  
 # 查找最后一个逗号的位置  
 last\_comma\_index = line.rfind(',')  
 # 如果找到了逗号，就截取逗号之前的部分  
 if last\_comma\_index != -1:  
 return line[:last\_comma\_index].rstrip() + '\n' # 确保去除末尾的空白字符并添加换行符  
 else:  
 # 如果没有找到逗号，返回原始行  
 return line  
def process\_file(input\_file\_path, output\_file\_path):  
 try:  
 with open(input\_file\_path, 'r', encoding='utf-8') as file:  
 content = file.read()  
 # 按行分割内容，然后处理每一行  
 lines = content.split('\n')  
 processed\_lines = [process\_line(line) for line in lines if line.strip()]  
 # 将处理后的行重新组合成一个字符串，并写入输出文件  
 with open(output\_file\_path, 'w', encoding='utf-8') as file:  
 file.write(''.join(processed\_lines))  
 print(f"文件处理完成，结果已保存到 {output\_file\_path}")  
 except Exception as e:  
 print(f"处理文件时发生错误: {e}")  
# 使用示例  
input\_file\_path = 'D:\\可视语言与信息可视化\\普通上机实验\\实验4\\3022207128-杨宇鑫-实验4\\实验4-2\\src\\vis\_paper\_citation\_network\\Nodes-New.csv' # 替换为你的输入文件路径  
output\_file\_path = 'D:\\可视语言与信息可视化\\普通上机实验\\实验4\\3022207128-杨宇鑫-实验4\\实验4-2\\src\\vis\_paper\_citation\_network\\Nodes-New1.csv' # 替换为你的输出文件路径  
process\_file(input\_file\_path, output\_file\_path)

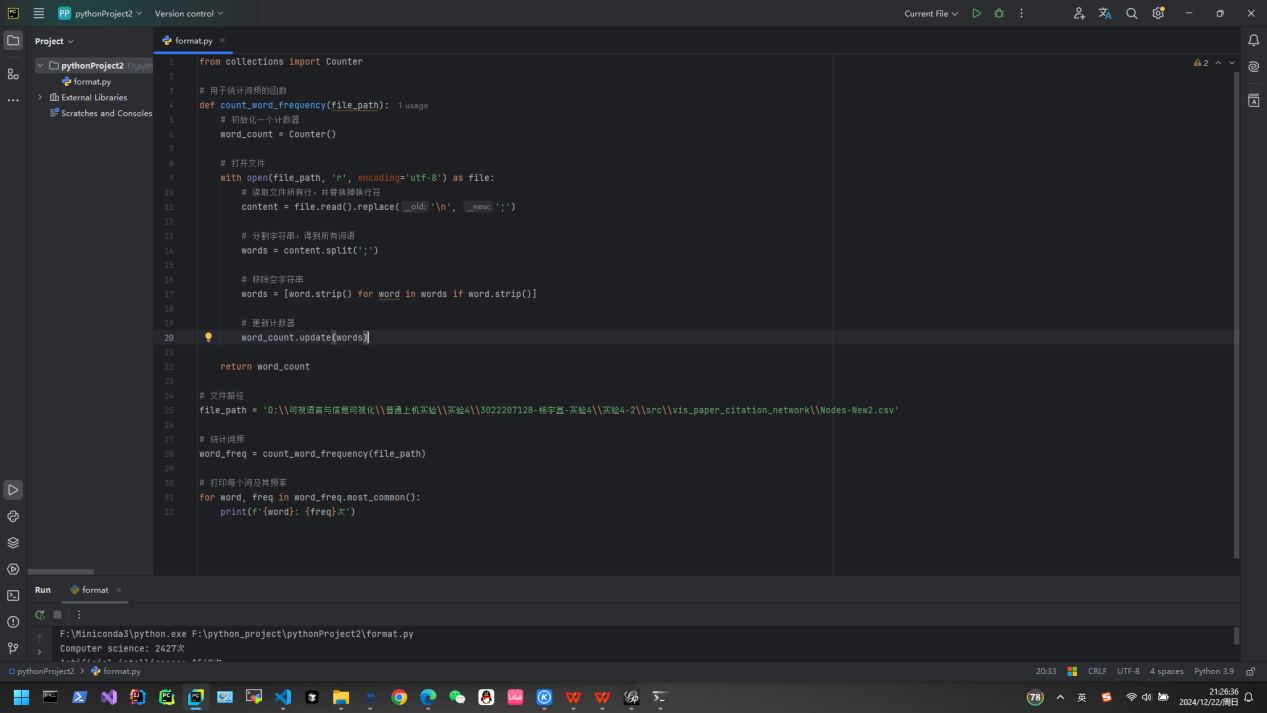


def process\_line(line):  
 # 查找最后一个逗号的位置  
 last\_comma\_index = line.rfind(',')  
 # 如果找到了逗号，就截取逗号之后的部分  
 if last\_comma\_index != -1:  
 return line[last\_comma\_index + 1:] # 包含逗号并返回逗号之后的内容  
 else:  
 # 如果没有找到逗号，返回原始行  
 return line  
def process\_file(input\_file\_path, output\_file\_path):  
 try:  
 with open(input\_file\_path, 'r', encoding='utf-8') as file:  
 content = file.read()  
 # 按行分割内容，然后处理每一行  
 lines = content.split('\n')  
 processed\_lines = [process\_line(line) for line in lines if line.strip()]  
 # 将处理后的行重新组合成一个字符串，并写入输出文件  
 with open(output\_file\_path, 'w', encoding='utf-8') as file:  
 file.write('\n'.join(processed\_lines))  
 print(f"文件处理完成，结果已保存到 {output\_file\_path}")  
 except Exception as e:  
 print(f"处理文件时发生错误: {e}")  
# 使用示例  
input\_file\_path = 'D:\\可视语言与信息可视化\\普通上机实验\\实验4\\3022207128-杨宇鑫-实验4\\实验4-2\\src\\vis\_paper\_citation\_network\\Nodes-New1.csv' # 替换为你的输入文件路径  
output\_file\_path = 'D:\\可视语言与信息可视化\\普通上机实验\\实验4\\3022207128-杨宇鑫-实验4\\实验4-2\\src\\vis\_paper\_citation\_network\\Nodes-New2.csv' # 替换为你的输出文件路径  
process\_file(input\_file\_path, output\_file\_path)



2.10.2 对只留下Fields\_of\_Study字段的数据进行词频统计

from collections import Counter  
# 用于统计词频的函数  
def count\_word\_frequency(file\_path):  
 # 初始化一个计数器  
 word\_count = Counter()  
 # 打开文件  
 with open(file\_path, 'r', encoding='utf-8') as file:  
 # 读取文件所有行，并替换掉换行符  
 content = file.read().replace('\n', ';')  
 # 分割字符串，得到所有词语  
 words = content.split(';')  
 # 移除空字符串  
 words = [word.strip() for word in words if word.strip()]  
 # 更新计数器  
 word\_count.update(words)  
 return word\_count  
# 文件路径  
file\_path = 'D:\\可视语言与信息可视化\\普通上机实验\\实验4\\3022207128-杨宇鑫-实验4\\实验4-2\\src\\vis\_paper\_citation\_network\\Nodes-New2.csv'  
# 统计词频  
word\_freq = count\_word\_frequency(file\_path)  
# 打印每个词及其频率  
for word, freq in word\_freq.most\_common():  
 print(f'{word}: {freq}次')



2.10.3 词频统计结果

一共3968个词语 , 由于词语过多 , 我将统计结果放在了文件中



下面展示前100条 :

Computer science: 2427次  
Artificial intelligence: 1549次  
Visualization: 1492次  
Computer vision: 1357次  
Data visualization: 1273次  
Theoretical computer science: 1056次  
Data mining: 679次  
Visual analytics: 408次  
Information visualization: 401次  
Rendering (computer graphics): 381次  
Mathematics: 315次  
Volume rendering: 268次  
Machine learning: 216次  
Interactive visualization: 213次  
User interface: 164次  
Computational geometry: 159次  
Cluster analysis: 152次  
Data set: 148次  
Computer graphics: 148次  
Human–computer interaction: 143次  
Scientific visualization: 141次  
Animation: 135次  
Mathematical optimization: 131次  
Flow visualization: 131次  
Vector field: 121次  
Feature extraction: 113次  
Computer graphics (images): 107次  
Data science: 107次  
Graph drawing: 103次  
Text mining: 99次  
Simulation: 98次  
Multimedia: 97次  
Data modeling: 95次  
Scalar (physics): 89次  
Scalability: 81次  
Isosurface: 77次  
Information retrieval: 77次  
Analytics: 77次  
Topology: 75次  
Data structure: 74次  
Polygon mesh: 74次  
Software: 69次  
Level of detail: 68次  
Interactive visual analysis: 68次  
Graphical user interface: 66次  
Mesh generation: 63次  
Discrete mathematics: 63次  
Grid: 62次  
Multivariate statistics: 62次  
Graph theory: 61次  
Parallel coordinates: 59次  
Graph: 58次  
Voxel: 57次  
Interpolation: 57次  
Image segmentation: 55次  
Scalar field: 54次  
Computational fluid dynamics: 54次  
Pattern recognition: 53次  
Interaction technique: 52次  
Computation: 52次  
Clustering high-dimensional data: 51次  
3D rendering: 49次  
Algorithm design: 49次  
Usability: 48次  
Image texture: 48次  
Computer animation: 47次  
Algorithm: 45次  
Real-time rendering: 45次  
Streamlines streaklines and pathlines: 45次  
Texture mapping: 44次  
Ray tracing (graphics): 43次  
Dimensionality reduction: 43次  
Graphics: 42次  
Ray casting: 42次  
Graphics hardware: 42次  
Geometric modeling: 40次  
Hierarchy: 39次  
Cognition: 39次  
Shading: 38次  
Vertex (geometry): 38次  
Filter (signal processing): 38次  
Workflow: 38次  
Creative visualization: 37次  
Transfer function: 37次  
Image processing: 37次  
Cultural analytics: 37次  
Polygon: 36次  
Iterative reconstruction: 36次  
Tensor: 36次  
Task analysis: 36次  
Pixel: 35次  
Social network: 34次  
Glyph: 34次  
Architecture: 33次  
Parallel rendering: 32次  
Time series: 32次  
Tiled rendering: 32次  
Diagram: 32次  
Software rendering: 32次  
Perception: 31次

2.11 对字段的词频统计的语义进行分析:

根据词频统计数据可以总结出以下语义信息，这些信息揭示了数据集中的主要研究领域和关注点：

**计算机科学（Computer Science）：**

数据集中最频繁出现的术语，表明计算机科学是核心研究领域。

涉及广泛的子领域，包括理论计算机科学、数据结构、算法设计、软件工程等。

**人工智能（Artificial Intelligence）：**

作为计算机科学的一个分支，人工智能在数据集中占有重要位置。

可能涉及机器学习、深度学习、自然语言处理等技术。

**可视化（Visualization）：**

包括数据可视化、信息可视化和科学可视化等多个方面。

强调了数据表示和交互式探索的重要性。

**计算机视觉（Computer Vision）：**

与图像和视频处理相关的技术，可能包括模式识别和图像理解。

与人工智能紧密相关，涉及特征提取、目标检测等。

**数据科学（Data Science）：**

数据挖掘和数据分析的频繁出现表明了对大数据和数据驱动方法的重视。

涉及数据集的预处理、特征工程、模型训练等。

**交互式可视化（Interactive Visualization）：**

用户界面和交互式设计的重要性，强调用户体验和参与度。

涉及人机交互、图形用户界面设计等。

**图形学（Computer Graphics）：**

包括渲染、3D渲染、纹理映射等技术，与计算机视觉和虚拟现实紧密相关。

强调了图形硬件和实时渲染技术的发展。

**数学（Mathematics）：**

数学在计算机科学和人工智能中的基础作用，特别是在算法开发和理论分析中。

涉及拓扑学、几何学、优化理论等。

**机器学习（Machine Learning）：**

作为人工智能的一个关键部分，机器学习在模式识别和预测模型中扮演重要角色。

涉及聚类分析、维度约减、算法设计等。

**并行计算（Parallel Computing）：**

并行处理和高性能计算的需求，特别是在大规模数据处理和复杂模拟中。

涉及计算几何、图形处理、数据结构的优化。

**流可视化（Flow Visualization）：**

流体动力学和矢量场的可视化，用于科学和工程领域的数据表示。

涉及计算流体动力学、矢量场分析等。

**多维数据（Multidimensional Data）：**

多维数据的挑战，如维度约减、聚类高维数据等。

涉及数据科学中的高级技术和方法。

这些语义信息揭示了数据集中的研究重点和技术趋势，反映了当前计算机科学和相关领域的研究热点。通过这些关键词，可以了解到研究者们在探索的技术前沿和面临的挑战。

2.12 结合语义和图谱中节点的分布写出我的发现

提供的词频统计数据和图谱中的节点分布可以发现几个关键点，这些关键点揭示了数据集中的研究主题和趋势。以下是我的发现：

**计算机科学的中心地位：**从词频统计中可以看出，“计算机科学”（Computer Science）是出现频率最高的词汇，这表明计算机科学是这个数据集中的核心领域。图谱中的节点分布也显示了这一点，因为与“计算机科学”相关的节点数量众多，且分布广泛。

**人工智能的广泛应用：**紧随其后的是“人工智能”（Artificial Intelligence），这表明人工智能是计算机科学领域中一个非常活跃和重要的分支。图谱中，与人工智能相关的节点形成了一个显著的集群组，这可能代表了人工智能在多个子领域中的应用。

**可视化的重要性：**词频中的“可视化”（Visualization）和“数据可视化”（Data visualization）以及“科学可视化”（Scientific visualization）等词汇频繁出现，强调了在计算机科学中，如何有效地表达和理解数据的重要性。图谱中的节点颜色和分布可能代表了不同的可视化技术或方法。

**数据科学与挖掘的兴起：**词频中的“数据科学”（Data Science）和“数据挖掘”（Data mining）显示出数据科学在现代研究中的重要性。图谱中的节点分布可能揭示了数据科学在多个领域的应用，如商业智能、健康信息学和社交网络分析等。

**跨学科的融合：**从词频中的“计算几何”（Computational geometry）、“数学”（Mathematics）、“图形学”（Graph theory）等可以看出，计算机科学与其他学科如数学、工程和自然科学的交叉融合。图谱中的节点分布可能显示了这些跨学科领域的联系和交互。

**用户界面与交互：**词频中的“用户界面”（User interface）和“人机交互”（Human–computer interaction）表明了在设计和开发过程中对用户体验的重视。图谱中的节点分布可能揭示了用户界面设计在不同应用中的重要性。

**高性能计算的需求：**词频中的“并行计算”（Parallel Computing）和“实时渲染”（Real-time rendering）等词汇的出现，反映了在处理大规模数据和复杂计算任务时对高性能计算的需求。图谱中的节点分布可能显示了高性能计算在多个领域的应用，如图形渲染、科学计算和工程模拟。

**机器学习的兴起：**词频中的“机器学习”（Machine Learning）和“特征提取”（Feature extraction）等词汇的频繁出现，表明了机器学习在数据分析和模式识别中的重要性。图谱中的节点分布可能揭示了机器学习算法在不同领域的应用，如图像识别、自然语言处理和推荐系统。

**图形和网络分析：**词频中的“图绘制”（Graph drawing）和“网络分析”（Network analysis）等词汇的出现，强调了图形和网络在数据表示和分析中的作用。图谱中的节点分布可能显示了图形和网络分析在社交网络、生物信息学和通信网络等领域的应用。

**模拟和动画的作用：**词频中的“模拟”（Simulation）和“动画”（Animation）表明了这些技术在研究和教育中的重要性。图谱中的节点分布可能揭示了模拟和动画在多个领域的应用，如教育、培训和娱乐。

这些发现为我们提供了一个全面的视角，以理解数据集中所包含的领域和主题。

通过分析词频和图谱中的节点分布，可以更好地理解当前的研究趋势和未来的发展方向。