Network Analysis for Digital Humanities

From scratch

让AI专业学生用Python（pandas+networkx）处理人文数据，完成"数据读取→清洗→网络构建→可视化"全流程，理解"节点/边"等技术术语在人文场景（历史人物关系）中的应用，为后续工程学习铺垫"人文数据应用"思维

Objectives: Equip AI-major students with Python (pandas + networkx) to process humanities data

Complete the full workflow of "data reading → cleaning → network construction → visualization".

Students will understand how technical terms like "nodes/edges" apply to humanities scenarios (historical figure relationships), laying the groundwork for bilingual "humanities data engineering applications" in the future.

Setup

In [15]:
```
# Check if required packages are installed (run once)
try:
    import pandas as pd
    import networkx as nx
    import matplotlib.pyplot as plt
    print("All required packages (pandas, networkx, matplotlib) are installed!")
except ImportError as e:
    missing_pkg = str(e).split("'")[1]
    print(f"X Missing package: {missing_pkg}")
    print("Run this in VS Code's terminal (venv activated):")
    print(f"pip install {missing_pkg}")
```

```
All required packages (pandas, networkx, matplotlib) are installed!
```

2. Key Terminology

Term	Explanation
DataFrame	A structure in Python for storing tabular data (e.g., nodes.csv becomes a DataFrame when imported)
Node	"Entities" in a network (e.g., "Voltaire" or "Rousseau" in our dataset)
Edge	"Relationships" between nodes (e.g., "Friend" or "Influenced by")
Network Graph	A visual representation of relationships using nodes and edges (our final output)
Package	Python tool libraries (pandas for data processing, networkx for network building, matplotlib for plotting)

3. Tool Intro

We'll use 3 tools. They are all basic Python libraries.

1. Pandas handles data reading
2. Networkx handles network linkages
3. Matplotlib handles chart plotting.

The total code is less than 50 lines."

In [16]:
```
# Import libraries: handle data processing, network building, and plotting respe
import pandas as pd  # pandas: processes tabular data (corresponds to DataFrame
import networkx as nx  # networkx: builds network structures
import matplotlib.pyplot as plt  # matplotlib: creates visualizations

# Font settings (ensure proper display of special characters)
plt.rcParams['font.sans-serif'] = ['DejaVu Sans']  # Font support
plt.rcParams['axes.unicode_minus'] = False  # Proper display of negative signs
```

This step is like 'grabbing tools'. As you would get pots and pans before cooking, here, we import libraries like pandas before processing data. It is convention (habit) to always rename the libraries when importing ('pd', 'nx', 'plt') This is both so that later commands can be typed in a short form, and (more importantly) because the 'import as' command means we will later add that short form to commands, ensuring no confusion when two libraries have very similar functions.

Example: matplotlib.pyplot and seaborn both have a plot() function. Using plt.plot() (instead of plot()) clarifies you're calling matplotlib's version, avoiding ambiguous behavior.

The font settings ensure all text displays correctly in our final visualization

In [17]:
```
# Read node data (figure information): specify CSV file path (modify based on yo
nodes = pd.read_csv('nodes.csv')
# Read edge data (relationship information)
edges = pd.read_csv('edges.csv')

# Check first 5 rows to confirm successful reading (humanities check: verify fig
print("Node data (first 5 rows): ")
print(nodes.head())
print("\nEdge data (first 5 rows): ")
print(edges.head())
```

```
Node data (first 5 rows):
   id         name       field
0   1     Voltaire  Philosophy
1   2     Rousseau  Philosophy
2   3  Montesquieu    Politics
3   4      Diderot  Literature
4   5   d'Alembert     Science

Edge data (first 5 rows):
   source  target   relationship
0       1       3         Friend
1       1       4   Collaborator
2       2       4         Critic
3       3       6  Influenced by
4       4       5      Coworkers
```

In [18]:
```
# 读取节点数据（人物信息）：指定CSV文件路径（需根据自己的文件位置修改！）
# Read node data (figure information): specify CSV file path (modify based on yo
nodes = pd.read_csv('nodes.csv')
# Read edge data (relationship information)
edges = pd.read_csv('edges.csv')

# Check first 5 rows to confirm successful reading (humanities check: verify fig
print("Node data (first 5 rows): ")
print(nodes.head())
print("\nEdge data (first 5 rows): ")
print(edges.head())
```

```
Node data (first 5 rows):
   id         name       field
0   1     Voltaire  Philosophy
1   2     Rousseau  Philosophy
2   3  Montesquieu    Politics
3   4      Diderot  Literature
4   5   d'Alembert     Science

Edge data (first 5 rows):
   source  target   relationship
0       1       3         Friend
1       1       4   Collaborator
2       2       4         Critic
3       3       6  Influenced by
4       4       5      Coworkers
```

运行后展示结果，讲解："pandas的read_csv把CSV文件变成了DataFrame——看nodes里有'伏尔泰'，edges里有'伏尔泰和孟德斯鸠是朋友'，如果数据读对了 You can see that the "pandas' read_csv" function converts CSV files into DataFrames—you can see 'Voltaire' in the nodes and 'Voltaire and Montesquieu are Friends' in the edges, that is, if the data reads correctly." 注意：若学生文件路径不同，提示修改 `'nodes.csv'` 为完整路径（如 `'C:/Users/XXX/Desktop/nodes.csv'`）。Note: If you have different file paths, you must modify `'nodes.csv'` to the full path (e.g., `'C:/Users/XXX/Desktop/nodes.csv'` on Windows or `'/Users/XXX/Desktop/nodes.csv'` on Mac).

Checks to Apply for All Data-Based Operations

In [19]:
```
# Check for missing values # 1. 检查缺失值（人文数据常缺漏，需清洗）

print("Missing values in node data: ")
print(nodes.isnull().sum())  # Output number of missing values per column (our d
print("\nMissing values in edge data: ")
print(edges.isnull().sum())

# 2. Filter invalid relationships (e.g., "self-relationships that occur in real
edges_clean = edges[edges['source'] != edges['target']]  # Keep only edges where
print("\nNumber of edges after cleaning: ", len(edges_clean), " (original number
```

```
Missing values in node data:
id       0
name     0
field    0
dtype: int64

Missing values in edge data:
source          0
target          0
relationship    0
dtype: int64

Number of edges after cleaning:  11  (original number:  11 )
```

人文数据很'脏'——比如历史文献里可能漏写人物领域，或误记关系。这里我们用一行代码过滤无效关系，这就是'计算思维'解决人文问题的关键。Humanities data is often 'messy'—historical documents might miss a figure's field or misrecord relationships. Here we use one line of code to filter invalid relationships. This is the key way computational thinking solves humanities problems.

Build the Network

In [20]:
```python
#Build the Network
# 1. 创建空网络
G = nx.Graph()  # Graph()表示无向网络（关系无方向，如"朋友"是双向的）Represents an u

# 2. 添加节点（从nodes数据框中提取id和name，关联领域信息） Add Nodes (Extract id and
for _, row in nodes.iterrows():  # 遍历每一行人物数据 Iterate through each line of
    G.add_node(
        row['id'],  # 节点唯一标识
        name=row['name'],  # 节点名称（人文信息）Node Unique Identifier
        field=row['field']  # 节点属性（领域，用于后续着色）Node Attributes (Domain
    )

#   趣味拓展：添加新节点（康德）和关系（仰慕伏尔泰）——即时展示网络扩展性Fun Expansion:
G.add_node(9, name='Kant', field='Philosophy')  # 新节点：id=9，康德，哲学领域（自
G.add_edge(9, 1, relationship='Admirer')  # 新关系：康德（9）仰慕伏尔泰（1）New Rel

# 3. 添加边（从清洗后的边数据中提取关系）
for _, row in edges_clean.iterrows():
    G.add_edge(
        row['source'],  # 源节点（谁）Source node (who)
        row['target'],  # 目标节点（和谁）target node (whom)
        relationship=row['relationship']  # 边属性（关系类型）Edge Attribute (Rela
    )

# 验证网络：查看节点和边数量（添加康德后应变为9节点、11边）
print("网络节点数：", G.number_of_nodes())  # 预期输出：9 Expected Output
print("网络边数：", G.number_of_edges())  # 预期输出：11 Expected Output
```

```
网络节点数： 9
网络边数： 11
```

特别注意我们加的康德：只需要两行代码，他就自动加入哲学阵营（红色节点），和伏尔泰建立了'仰慕'关系——这就是网络分析的魔力，新增数据能即时融入并展示新关联！This is the core step—we convert 'figures' in the table to 'nodes' in the network, 'relationships'

to 'edges', and add 'field' attributes to nodes (Philosophy/Economics) and 'relationship type' attributes to edges, to show the network's historical interpretive meaning.

这和你们未来学的图神经网络（GNN）的节点/边构建逻辑完全一致，只是这里的属性是‘人文标签’而非‘特征向量’。Note that this is the same logic as node/edge construction in Graph Neural Networks (GNNs) which you'll use in other AI applications, except here the attributes are 'humanities labels' instead of 'feature vectors'

Visualize the Network (like formatting a document)

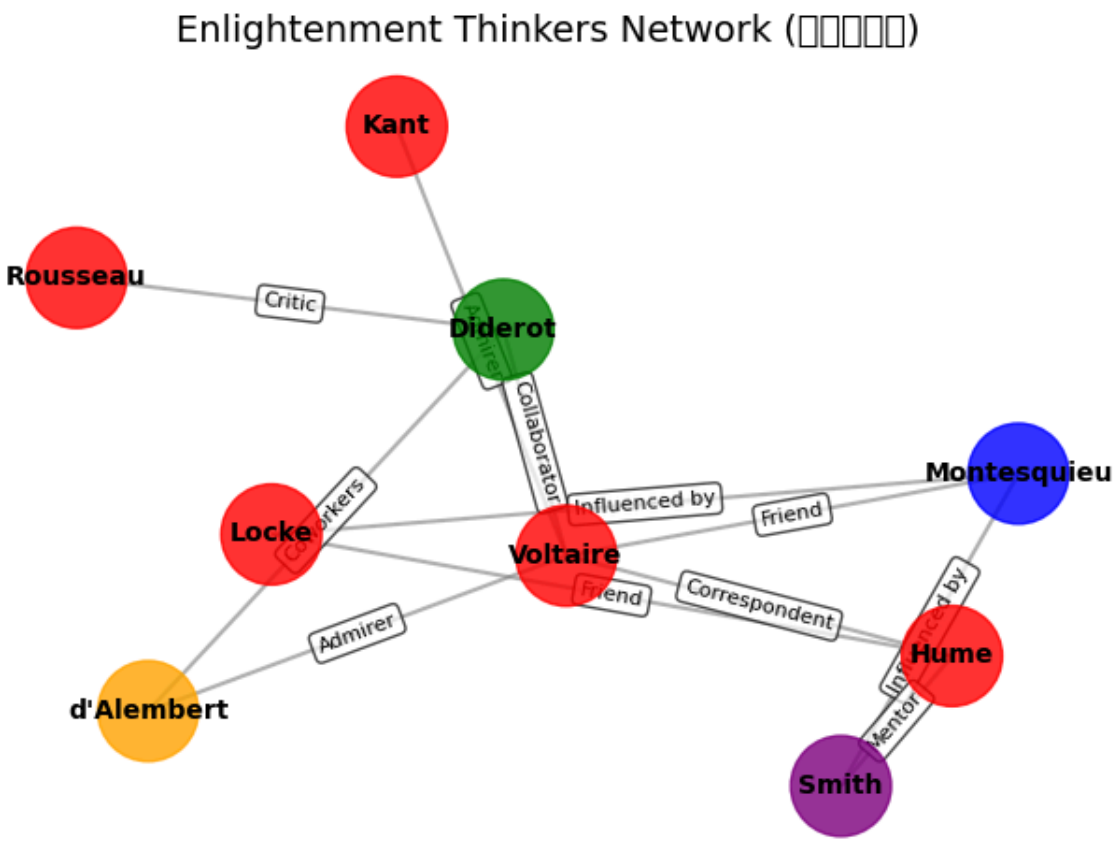
Randomness & Network Analysis

If you run this program several times you will notice that your network layout changes every time. There is nothing wrong with your program, this is just due to the the fact that the nx.spring_layout() function that is used to position the nodes in your diagram uses a stochastic (random) algorithm called the Fruchterman-Reingold force directed algorithm. What the algorithm does is makes nodes behave like particles with repulsion, so they push each other apart, and edges behave like springs with attraction, pulling connected nodes together. It starts in a random initial position for each node each time the program is run. It iterates (repeats) to find a 'balanced' layout, but the random starting point means the final positions are slightly different every time. Importantly, though, the edges and nodes are fixed (Voltaire is still friends with Montesquieu and Kant still admires Voltaire). It is only the visual appearance of the network that changes, for instance, one time grouping people on the right, another time on the left, rather than changing any of the relationships. If you want your layout to stay the same every time you run the program, add a 'seed' after the k value, i.e., (G, k=2, seed=42)

In [ ]:
```
# 1. 设置绘图布局（让网络图更美观，spring_layout是经典布局）
# 1. Set the drawing layout (to make the network graph more aesthetically pleasi
pos = nx.spring_layout(G, k=2, iterations=50)  # k控制节点间距，iterations控制迭代


# 2. 给不同领域的节点分配颜色（人文属性可视化） Assign colors to nodes in different
field_colors = {
    'Philosophy': 'red',    # 哲学：红色
    'Politics': 'blue',     # 政治：蓝色
    'Literature': 'green',  # 文学：绿色
    'Science': 'orange',    # 科学：橙色
    'Economics': 'purple'   # 经济学：紫色
}
# 提取每个节点的颜色（根据field属性匹配，康德自动匹配哲学红色）
# Take the color of each node (matched according to the field attribute, Kant au
node_colors = [field_colors[G.nodes[node]['field']] for node in G.nodes]

# 3. 绘制节点 Draw Nodes
nx.draw_networkx_nodes(
    G, pos,  # 网络和布局 Network & Layout
    node_color=node_colors,  # 节点颜色（领域区分）Node Color (Domain Differentiat
    node_size=1500,  # 节点大小 Node Size
    alpha=0.8  # 透明度 Transparency
)
```

```
# 4. 绘制边
nx.draw_networkx_edges(
    G, pos,
    edge_color='gray',  # 边颜色
    width=1.5,  # 边宽度
    alpha=0.6  # 透明度
)

# 5. 绘制节点标签（显示人物名字，而非id）Draw node labels (displaying character nam
node_labels = {node: G.nodes[node]['name'] for node in G.nodes}
nx.draw_networkx_labels(
    G, pos,
    labels=node_labels,  # 标签为人物名字（含康德）Tags are labeled with personal
    font_size=10,
    font_weight='bold'
)

# 6. 绘制边标签（可选：注释/取消注释切换“简洁版”和“详细版”）
#    演示时可先注释（简洁），再取消注释（显示关系），对比效果
# 6. Draw edge labels (optional: toggle between 'Simplified' and 'Detailed' by c

#    For demonstration, you can first comment out (simplified) and then uncommen
edge_labels = nx.get_edge_attributes(G, 'relationship')
nx.draw_networkx_edge_labels(
    G, pos,
    edge_labels=edge_labels,
    font_size=8,
    bbox=dict(boxstyle='round,pad=0.3', fc='white', alpha=0.7)  # 标签背景 Tag Ba
)

# 7. Adjust layout and display调整布局并显示
plt.axis('off')  # 隐藏坐标轴 Hide the coordinate axis
plt.title('Enlightenment Thinkers Network (含康德拓展)', fontsize=14)  # 更新标题
plt.tight_layout()  # 调整布局 Adjust the Layout
plt.show()  # 显示图表 Show chart
```

```
C:\Users\geronimo stilton\AppData\Local\Temp\ipykernel_27676\360105773.py:59: Use
rWarning: Glyph 21547 (\N{CJK UNIFIED IDEOGRAPH-542B}) missing from font(s) DejaV
u Sans.
  plt.tight_layout()  # 调整布局 Adjust the Layout
C:\Users\geronimo stilton\AppData\Local\Temp\ipykernel_27676\360105773.py:59: Use
rWarning: Glyph 24247 (\N{CJK UNIFIED IDEOGRAPH-5EB7}) missing from font(s) DejaV
u Sans.
  plt.tight_layout()  # 调整布局 Adjust the Layout
C:\Users\geronimo stilton\AppData\Local\Temp\ipykernel_27676\360105773.py:59: Use
rWarning: Glyph 24503 (\N{CJK UNIFIED IDEOGRAPH-5FB7}) missing from font(s) DejaV
u Sans.
  plt.tight_layout()  # 调整布局 Adjust the Layout
C:\Users\geronimo stilton\AppData\Local\Temp\ipykernel_27676\360105773.py:59: Use
rWarning: Glyph 25299 (\N{CJK UNIFIED IDEOGRAPH-62D3}) missing from font(s) DejaV
u Sans.
  plt.tight_layout()  # 调整布局 Adjust the Layout
C:\Users\geronimo stilton\AppData\Local\Temp\ipykernel_27676\360105773.py:59: Use
rWarning: Glyph 23637 (\N{CJK UNIFIED IDEOGRAPH-5C55}) missing from font(s) DejaV
u Sans.
  plt.tight_layout()  # 调整布局 Adjust the Layout
```



In [ ]:

4-5人一组，每组1台有Python环境的电脑（无环境的组看演示，或用教师电脑轮换）。任务：“复现教师演示的网络图，然后完成1个小修改（二选一）。In groups of 4-5, with one computer per group that has the Python environment set up. Students without access to the environment can watch the demonstration or take turns using the teacher's computer. Task: Replicate the network graph from the teacher's demonstration, then complete one small modification (choose one of two options).

TROUBLESHOOTING

- 问题1：中文乱码 → 检查 `plt.rcParams` 设置是否正确。
- 问题2：文件找不到 → 修改CSV文件路径为绝对路径。
- 问题3：依赖包未安装 → 指导运行 `pip install pandas networkx matplotlib` 。
- Issue 1: Font display errors → Check that `plt.rcParams` settings are correct.
- Issue 2: File not found → Modify the CSV file path to the full path.
- Issue 3: Missing packages → Guide students to run `pip install pandas networkx matplotlib` in the command line.

1. 拓展任务（二选一）| Extension Task (15 Mins)3. Extension Task (Choose One)

modifying code = changing humanities interpretation":

1. 属性修改 | Attribute Modification：给nodes.csv加1列“country”（国家，如Voltaire→France，Locke→UK），修改代码让节点颜色按“国家”区分，观察“国家与思想传播的关系” Attribute Modification: Add a "country" column to nodes.csv (e.g., Voltaire→France, Locke→UK, Hume→UK, Smith→UK, Montesquieu→France, Rousseau→France, Diderot→France, d'Alembert→France). Modify the code to color nodes by "country" and observe "national connections in Enlightenment thought传播"
2. 关系筛选 | Relationship Filtering：只保留“Friend”（朋友）关系的边，重新绘图，观察“启蒙运动的核心朋友圈”（伏尔泰-孟德斯鸠-洛克-休谟）。Relationship Filtering: Keep only edges with "Friend" relationships and redraw the graph. Observe the "core social circle of the Enlightenment" (Voltaire-Montesquieu-Locke-Hume).