张亦晴 3120235334

第一个数据集：

来源： Microsoft 资讯推荐 https://learn.microsoft.com/zh-cn/azure/open-datasets/dataset-microsoft-news?tabs=azureml-opendatasets

本数据集关于Microsoft新闻关系分析

第一个数据集：

来源： SNAP(Stanford Large Network Dataset Collection): http://snap.stanford.edu/data/index.html

Social circles: Facebook https://snap.stanford.edu/data/ego-Facebook.html

本数据集关于Facebook用户关系分析

In [3]:
```
import pandas as pd
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt
from random import randint

%matplotlib inline
```

数据获取与预处理

In [4]:
```
facebook = pd.read_csv(
    "/kaggle/input/snapdataset/facebook_combined.txt",
    #compression="gzip",
    sep=" ",
    names=["start_node", "end_node"],
)
facebook
```

Out[4]:

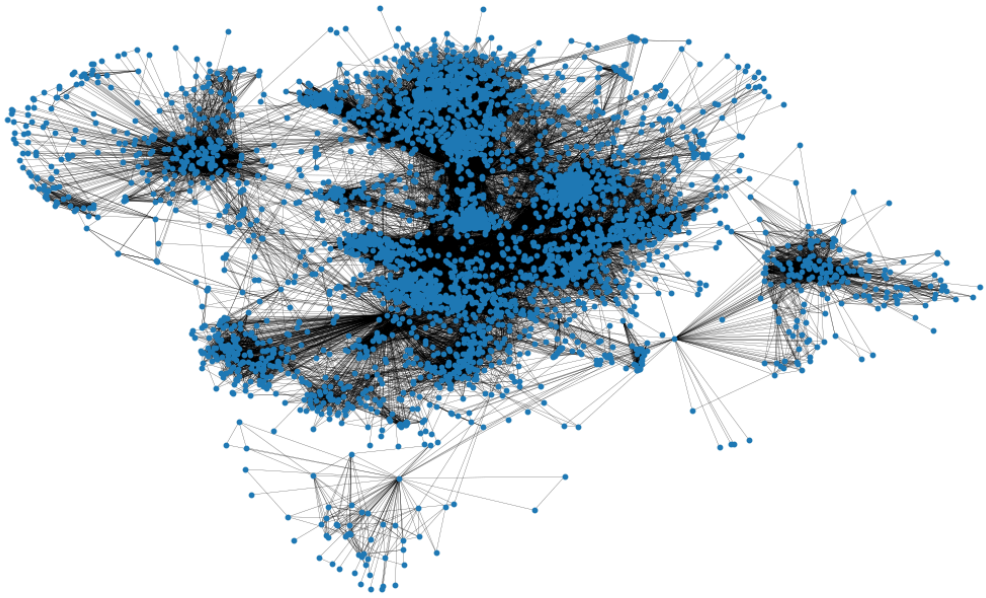
	start_node	end_node
0	0	1
1	0	2
2	0	3
3	0	4
4	0	5
...	...	...
88229	4026	4030
88230	4027	4031
88231	4027	4032
88232	4027	4038
88233	4031	4038

88234 rows × 2 columns

数据集可视化展示如下：

In [5]:
```
G = nx.from_pandas_edgelist(facebook, "start_node", "end_node")
```

In [7]:
```
plot_options = {"node_size": 10, "with_labels": False, "width": 0.15}
pos = nx.spring_layout(G, iterations=15, seed=1721)
fig, ax = plt.subplots(figsize=(15, 9))
ax.axis("off")
nx.draw_networkx(G, pos=pos, ax=ax, **plot_options)
```



In [8]:
```
connections_matrix = nx.adjacency_matrix(G)
dataset = connections_matrix.todense()
dataset
```

Out[8]:
```
array([[0, 1, 1, ..., 0, 0, 0],
       [1, 0, 0, ..., 0, 0, 0],
       [1, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]])
```

In [9]:
```
num_cols = dataset.shape[1]  # 获取列数
items = [str(i+1) for i in range(num_cols)]

# 转换函数
def array_to_transactions(data, items):
    num_rows, num_cols = data.shape  # 获取数组的维度
    dataset = []
    for row in range(num_rows):
        transaction = []
        for col in range(num_cols):
            if data[row, col] == 1:
                transaction.append(items[col])
        dataset.append(transaction)
    return dataset

# 调用函数
data = array_to_transactions(dataset, items)
```

频繁模式挖掘：社交网络的关系

In [12]:
```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori

# dataset = facebook
te = TransactionEncoder()
te_ary = te.fit(data).transform(data)
df = pd.DataFrame(te_ary, columns=te.columns_)
df
```

Out[12]:

	1	10	100	1000	1001	1002	1003	1004	1005	1006	...	990	991	992	993
0	False	True	True	False	False	False	False	False	False	False	...	False	False	False	False
1	True	False	False	False	False	False	False	False	False	False	...	False	False	False	False
2	True	False	False	False	False	False	False	False	False	False	...	False	False	False	False
3	True	True	False	False	False	False	False	False	False	False	...	False	False	False	False
4	True	False	False	False	False	False	False	False	False	False	...	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4034	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False
4035	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False
4036	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False
4037	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False
4038	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False

4039 rows × 4039 columns

In [13]:
```
frequent_itemsets = apriori(df, min_support=0.05, use_colnames=True)
frequent_itemsets['length'] = frequent_itemsets['itemsets'].apply(lambda x: len(x))
frequent_itemsets
```

Out[13]:

	support	itemsets	length
0	0.085912	(1)	1
1	0.052241	(1071)	1
2	0.050755	(1076)	1
3	0.258727	(108)	1
4	0.058183	(1150)	1
...	...	...	...
68	0.051003	(353, 3446)	2
69	0.052241	(1286, 108, 1374)	3
70	0.058183	(353, 2155, 1491)	3
71	0.052241	(353, 2146, 3107)	3
72	0.050260	(3147, 353, 2146)	3

73 rows × 3 columns

频繁模式挖掘分析

In [33]:
```
from mlxtend.frequent_patterns import association_rules

ar = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.02)
ar
```

Out[33]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage
0	(108)	(1071)	0.258727	0.052241	0.051993	0.200957	3.846754	0.038477
1	(1071)	(108)	0.052241	0.258727	0.051993	0.995261	3.846754	0.038477
2	(108)	(1076)	0.258727	0.050755	0.050508	0.195215	3.846218	0.037376
3	(1076)	(108)	0.050755	0.258727	0.050508	0.995122	3.846218	0.037376
4	(108)	(1150)	0.258727	0.058183	0.057935	0.223923	3.848625	0.042882
...	...	...	...	...	...	...	...	...
85	(3147, 2146)	(353)	0.050508	0.186927	0.050260	0.995098	5.323445	0.040819
86	(353, 2146)	(3147)	0.057688	0.051993	0.050260	0.871245	16.756938	0.047261
87	(3147)	(353, 2146)	0.051993	0.057688	0.050260	0.966667	16.756938	0.047261
88	(353)	(3147, 2146)	0.186927	0.050508	0.050260	0.268874	5.323445	0.040819
89	(2146)	(3147, 353)	0.057935	0.051745	0.050260	0.867521	16.765162	0.047262

90 rows × 10 columns

In [17]:
```
u_set = set()
for it in frequent_itemsets['itemsets']:
    u_set = u_set.union(it)
```

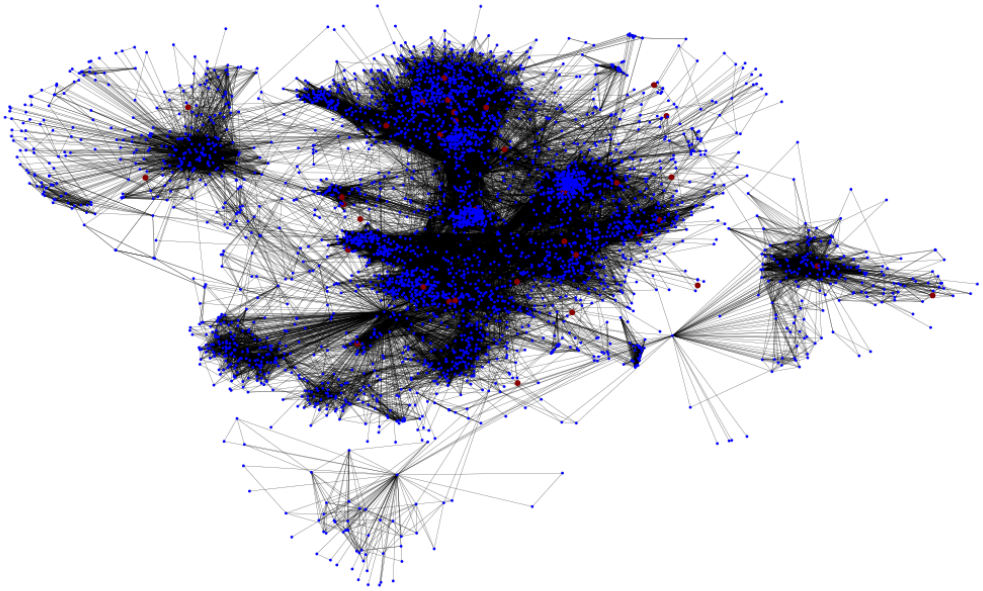
In [30]:
```
u_set2 = set()
for it in ar['antecedents']:
    u_set2 = u_set2.union(it)
for it in ar['consequents']:
    u_set2 = u_set2.union(it)
u_set2
```

Out[30]:
```
{'1071',
 '1076',
 '108',
 '1150',
 '1216',
 '1254',
 '1286',
 '1374',
 '1491',
 '2127',
 '2131',
 '2146',
 '2155',
 '2996',
 '3003',
 '3059',
 '3097',
 '3107',
 '3147',
 '3158',
 '3166',
 '3170',
 '3297',
 '3446',
 '353',
 '573',
 '686',
 '839',
 '918'}
```

In [59]:
```
color_map = []
sizes = []
for node in G:
    if str(node) in u_set:
        color_map.append('#8B0000')
        sizes.append(10)
    else:
        color_map.append('#0000FF')
        sizes.append(1)
```

```
'1'
'108'
'3003'
'353'
'918'
'1071'
'1076'
'1150'
'1216'
'1232'
'1254'
'1286'
'1374'
'1491'
'1822'
'2127'
'349'
'352'
'367'
'573'
'686'
'839'
'3097'
'2996'
'3059'
'3107'
'3147'
'3158'
'3166'
'3170'
'3297'
'3339'
'2155'
'2131'
'2146'
'3446'
```

In [60]:
```
plot_options = {"with_labels": False, "width": 0.15}
pos = nx.spring_layout(G, iterations=15, seed=1721)
fig, ax = plt.subplots(figsize=(15, 9))
ax.axis("off")
nx.draw_networkx(G, pos=pos, ax=ax, node_color=color_map, node_size=sizes, **plot_op
```



In [ ]:

In [ ]: