Finding Groups of Related Keywords

After running computeKeyWordsMatrix.py and entering the interactive mode, do the following:

>>> B = (MIK > 2e-3)

>>> B = 1\*B

The last command is to convert the Boolean values in B to ‘0’s and ‘1’s.

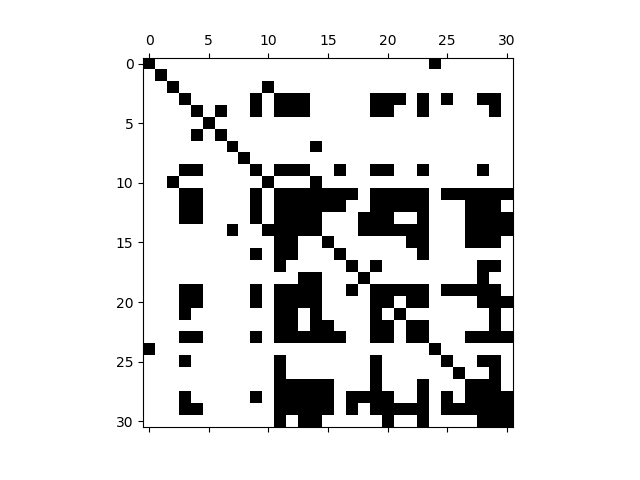
We can visualize which pairs of words has mutual information larger than 2e-3, by using the spy() function in matplotlib.

>>> import matplotlib.pyplot as plt

>>> plt.spy(B)

>>> plt.show()

This will look like:



We see indeed there are groups of words that are more related to each other than they are to words outside of the groups. To discover these groups, we perform complete-linkage hierarchical clustering on B as

>>> from scipy.cluster.hierarchy import dendrogram, linkage

>>> Z = linkage(B, method='complete', metric='cosine')

The output is

{'icoord': [[25.0, 25.0, 35.0, 35.0], [15.0, 15.0, 30.0, 30.0], [5.0, 5.0, 22.5, 22.5], [65.0, 65.0, 75.0, 75.0], [55.0, 55.0, 70.0, 70.0], [95.0, 95.0, 105.0, 105.0], [85.0, 85.0, 100.0, 100.0], [135.0, 135.0, 145.0, 145.0], [155.0, 155.0, 165.0, 165.0], [140.0, 140.0, 160.0, 160.0], [195.0, 195.0, 205.0, 205.0], [185.0, 185.0, 200.0, 200.0], [175.0, 175.0, 192.5, 192.5], [150.0, 150.0, 183.75, 183.75], [125.0, 125.0, 166.875, 166.875], [115.0, 115.0, 145.9375, 145.9375], [92.5, 92.5, 130.46875, 130.46875], [62.5, 62.5, 111.484375, 111.484375], [45.0, 45.0, 86.9921875, 86.9921875], [13.75, 13.75, 65.99609375, 65.99609375], [215.0, 215.0, 225.0, 225.0], [235.0, 235.0, 245.0, 245.0], [255.0, 255.0, 265.0, 265.0], [295.0, 295.0, 305.0, 305.0], [285.0, 285.0, 300.0, 300.0], [275.0, 275.0, 292.5, 292.5], [260.0, 260.0, 283.75, 283.75], [240.0, 240.0, 271.875, 271.875], [220.0, 220.0, 255.9375, 255.9375], [39.873046875, 39.873046875, 237.96875, 237.96875]], 'dcoord': [[0.0, 0.26970325665977857, 0.26970325665977857, 0.0], [0.0, 0.38762756430420542, 0.38762756430420542, 0.26970325665977857], [0.0, 0.49290744716289014, 0.49290744716289014, 0.38762756430420542], [0.0, 0.21737620787507361, 0.21737620787507361, 0.0], [0.0, 0.29289321881345254, 0.29289321881345254, 0.21737620787507361], [0.0, 0.21665054819935969, 0.21665054819935969, 0.0], [0.0, 0.26970325665977857, 0.26970325665977857, 0.21665054819935969], [0.0, 0.058823529411764719, 0.058823529411764719, 0.0], [0.0, 0.10291477285493944, 0.10291477285493944, 0.0], [0.058823529411764719, 0.18590842158930576, 0.18590842158930576, 0.10291477285493944], [0.0, 0.04881026878865824, 0.04881026878865824, 0.0], [0.0, 0.080745280259012286, 0.080745280259012286, 0.04881026878865824], [0.0, 0.19967326933495877, 0.19967326933495877, 0.080745280259012286], [0.18590842158930576, 0.23529411764705888, 0.23529411764705888, 0.19967326933495877], [0.0, 0.29289321881345243, 0.29289321881345243, 0.23529411764705888], [0.0, 0.41666666666666663, 0.41666666666666663, 0.29289321881345243], [0.26970325665977857, 0.47235514698891368, 0.47235514698891368, 0.41666666666666663], [0.29289321881345254, 0.57359856728877912, 0.57359856728877912, 0.47235514698891368], [0.0, 0.69220649437445381, 0.69220649437445381, 0.57359856728877912], [0.49290744716289014, 0.81742581416494464, 0.81742581416494464, 0.69220649437445381], [0.0, 2.2204460492503131e-16, 2.2204460492503131e-16, 0.0], [0.0, 0.64644660940672627, 0.64644660940672627, 0.0], [0.0, 0.18350341907227408, 0.18350341907227408, 0.0], [0.0, 1.0, 1.0, 0.0], [0.0, 1.0, 1.0, 1.0], [0.0, 1.0, 1.0, 1.0], [0.18350341907227408, 1.0, 1.0, 1.0], [0.64644660940672627, 1.0, 1.0, 1.0], [2.2204460492503131e-16, 1.0, 1.0, 1.0], [0.81742581416494464, 1.0, 1.0, 1.0]], 'ivl': ['21', '26', '17', '25', '16', '22', '15', '27', '4', '3', '9', '30', '14', '12', '23', '13', '20', '28', '19', '11', '29', '0', '24', '7', '18', '2', '10', '8', '6', '1', '5'], 'leaves': [21, 26, 17, 25, 16, 22, 15, 27, 4, 3, 9, 30, 14, 12, 23, 13, 20, 28, 19, 11, 29, 0, 24, 7, 18, 2, 10, 8, 6, 1, 5], 'color\_list': ['g', 'g', 'g', 'r', 'r', 'r', 'r', 'r', 'r', 'r', 'r', 'r', 'r', 'r', 'r', 'r', 'r', 'r', 'r', 'b', 'c', 'm', 'y', 'b', 'b', 'b', 'b', 'b', 'b', 'b']}

In this output, we want to save ‘leaves’ as kS:

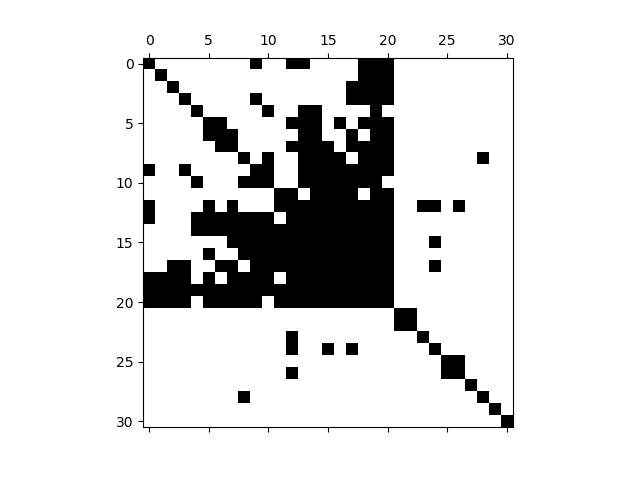
>>> kS = [21, 26, 17, 25, 16, 22, 15, 27, 4, 3, 9, 30, 14, 12, 23, 13, 20, 28, 19, 11, 29, 0, 24, 7, 18, 2, 10, 8, 6, 1, 5]

Having this, we can check whether hierarchical clustering has grouped the keywords correctly, by showing the re-ordered matrix B as

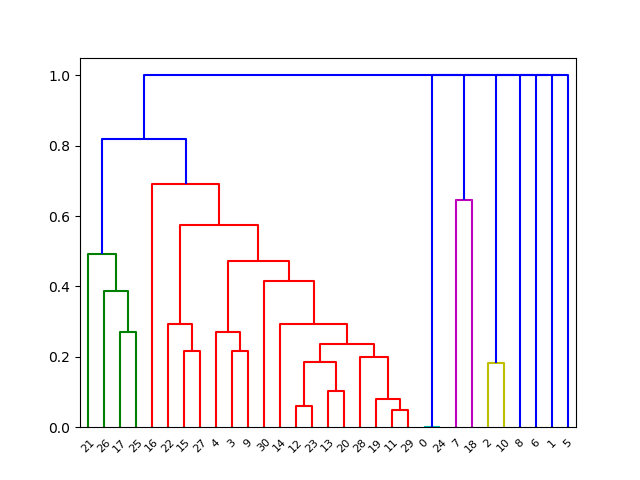
>>> plt.spy(B[kS, :][:, kS])

>>> plt.show()

The re-ordered matrix looks like:



The blocks corresponds to clusters in the dendrogram shown below:



where the top block starts from keyword #21 and ends at keyword #29. The other non-trivial clusters are {0, 24} and {2, 10}.

The top block consists of words like

'petition', 'says', 'coop', 'restrict', 'bin', 'ram', 'ask',

'tempt', 'Fin', 'Euro', 'Reuter', 'weeks', 'analyst', 'able', 'ran',

'aft', 'owing', 'trade', 'many', 'Th', 'use'

which seems to be related to trade. Then again, Reuters report a lot of financial and business news.

You may now repeat this procedure for different thresholds.