MLSD: Assignment 3
Community Detection in Social Networks
Streams

– Due date: June 6, 2023 –

1. Compare the following implementations of Spectral Graph Partitioning for community detection:

   1.1 Calculate the eigendecomposition of the graph Laplacian matrix and cluster the nodes of the network using a low rank representation consisting of the first n eigenvectors.
   You can use the eigengap approach to determine the number of clusters. The eigengap is the difference between successive eigenvalues, sorted in ascending order; you should consider the first peak in the eigengap as the optimal number of clusters.

   1.2 Use the Scikit-learn Spectral Clustering implementation.

   1.3 Use the Spark Power Iteration Clustering implementation.

   Apply all the methods to the following datasets:

   - Facebook
   - Phenomenology collaboration network
   - Human protein-protein interaction network

   Notes:
   You can use Python libraries, including Numpy, Scipy, NetworkX, scikit-learn, to implement each step of the spectral partitioning method.

2. Streams

   In these exercises you should make use of Spark Streaming, preferably the Structured Streaming engine: Structured Streaming Programming Guide
   The use of Spark Streaming is mandatory in exercise 2.1 and recommended in exercise 2.1, in which a small penalty will be applied to the grade if streaming is not used.

   You may use the code provided (‘simple_socket_server.py’) as a starting point to generate the data to be consumed by the streaming engine.

   2.1 Implement the DGIM method to estimate the number of 1s within a window of size $k \leq N$, where k and N are parameters. Test it by generating a synthetic bit stream and estimate the number of 1s in the stream at user defined intervals. You should also show the correct number of 1s, which is known.

2.2 Using the streaming dataset provided, apply the exponentially decaying window approach to keep smoothed counts of occurring events. Inspect the stream at one second intervals (to check which events occurred), and display the 5 most frequent events at (user-defined) intervals of t seconds.