Mini-project #1:

Using community detection for malware detection

# Brief summary

In this mini-project you will generate networks based on the dataset. You will then apply community detection algorithms to these networks. Using features derived from these communities, you will train a classifier whose goal is to detect malicious files.

This short guide is a based on [this](https://www.sciencedirect.com/science/article/pii/S0950705117305336?via%3Dihub) paper, sections 4.3 + 4.4.   
Make sure you read and understand the paper.

# Generating the networks

Using all the data from the first 5 days, construct one of the following 4 networks (we recommend you start with files and domains).

|  |  |  |
| --- | --- | --- |
| **Nodes** | **Edges** | **Weights** |
| Files and domains | File was downloaded from domain | Number of machines file was downloaded to from this domain |
| Files and machines | File was seen on machine | All 1 |
| Files and hostnames | File was downloaded from hostname | Number of machines file was downloaded to from this hostname |
| Machines and domains | Machine downloaded a file from domain | Number of files downloaded from the domain to the machine |

The data from the last 2 days will be used for testing the model you build based on the first 5 days, so you should build these networks also based on the data of the last two days.

Analyze the networks to better understand them. For example, answer questions like: what is the average degree? Does one type of nodes behave differently from the other? How does the degrees distribution look like? Consider removing high/low degree nodes.

Extra: consider performing the analysis for an additional graph from the list.

*Include in your report the graphs and statistics of the analysis you perform and describe any insights you gained.*

# Run community detection algorithm(s)

On each of the networks, run a community detection algorithm - there are many options, such as [[1]](https://en.wikipedia.org/wiki/Louvain_Modularity), [[2]](https://sites.google.com/site/greedycliqueexpansion/) and more. Start with a simple non-weighted, non-overlapping algorithm.

Extra: consider using more complex ones:

* 1. weighted/unweighted
  2. overlapping/non-overlapping
  3. Online algorithm (updates when nodes and edges are added)

*Include in your report information about each of the algorithms you used, a high-level description of its logic, its pros and cons, its run-time (theoretical and according to measurements you perform) etc.*

Checkpoint

# Communities analysis

Analyze the properties of resulting communities, such as:

1. Community size distribution
2. “[Quality](https://arxiv.org/pdf/1510.01714.pdf)” of communities
3. Malicious rate distribution

Etc.

*Include in your report the graphs and statistics of the analysis you perform, and describe any insights you gained.*

# Feature extraction

Per file, extract both “global” features (size, prevalence, number of source domains etc.) and features from the communities to which it belongs (community size, malicious/total amount of nodes, etc.).

Analyze per feature its distribution, comparing clean vs malicious files. Plot them in a manner that will show the difference/resemblance between them (TIP: you may want to exclude extreme values).

*Include in your report the graph and statistics of the analysis you perform, and describe any insights you gained, like what feature seems more helpful, why the distribution is as it is etc.*

Checkpoint

# Machine learning

Note that our data contains only malicious labels. As for clean files, use files which are common and prevalent in the data.

1. Use the features extracted to classify files to clean and malicious. Make sure you use two (or more) types of classifiers and select the most appropriate one.
2. Use cross-validation to check your results and select the model
3. Make sure to use also a time-split to test your results. Be careful with what files you test yourself on. The files should only appear in the last two days (otherwise they are part of the training data).

*Include in your report the graph and statistics of the results of the classification. Make sure you compare the models you use. Do the results from the cross-validation align with those of the time-split?*

Checkpoint

# Bonus – dealing with modularity

Modularity is a metric used to estimate the quality of the partition of the nodes to communities.

As defined in [Newman’s work](http://www.pnas.org/content/103/23/8577.long), given a partition of the nodes *C*, the modularity of the partition is:

Where A is the adjacency matrix, and is the degree of node i.

Note that this formula does not support weights, nor overlapping communities.

1. Add support to weights. Make sure your formula reduces to the non-weighted version when all weights are 1.
2. Add support to overlapping communities. Make sure your formula reduces to the non-overlapping version when nodes belong to a single community.