Social Network Analysis
Final Project

Giulio Rossetti, Valentina Pansanella
giulio.rossetti@isti.cnr.it, valentina.pansanella@sns.it

Introduction

The Final Project composes of four parts:

- Data Collection,
- Network Analysis,
- Open Question, and
- Activity Report

The details and constraints of each part are described in this document.

General Rules

1. Groups should be composed of, at most, 4 students;
2. All students need to have a GitHub (`https://github.com/`) account;
3. Every group must access and register to the following GitHub classroom repository: `https://classroom.github.com/a/PWIDvGx6`
4. Code, data, and report must be uploaded on such repository.

☞ **General rules:** When first accessing the GitHub classroom it will be required to form a new team to which add your colleagues as members.

When selecting the group name - which will identify the repository's name - follow the pattern

2023_surname1_surname2_surname3_surname4

Before starting the project, send an email to the course instructor specifying: name surname and student id of all group members (along with the planned data source).

Part 1: Data Collection

Data collection can be carried out without any restriction on programming languages (python is only a warm suggestion) and online sources.

☞ **Warning:** Using pre-built network datasets available on dedicated repositories (e.g., networkrepository, socialcomputing, snap, konect...) is not allowed.

It is allowed to build network starting from any kind of data (e.g., structured - database, API call results... - or unstructured - data dumps, text...).

Workflow

1. Identify an online data source,
2. Identify the entities (nodes) and relationships among them (edges),
3. Identify the available additional information to be collected (e.g., nodes' attributes, edge weights...),
4. Obtain the data from the selected data source (through API - if available - or by crawling),
5. Build a network from the data!

Requirements

- The network must have *at least 10-15k nodes.* Specific cases involving the analysis of smaller networks must be discussed beforehand with the instructors.
- The produced code **must** be made available on the group's GitHub project along with a brief description of the choices made/strategies adopted to perform data collection.
- The final version of the data (i.e., the network and, if present, all additional data) **must** be compressed and made available within the same folder in the GitHub repository.

Data Sources ideas

Last.fm, Blogs, Reddit, Blabla car, Linkedin, Wikipedia Corpora, Newspaper...

Part 2: Network Analysis

The analysis can be performed either by using a visual tool (i.e., Cytoscape and Gephi) and/or the by means of a programming language. The use of python (networkx or igraph) is not mandatory, although, strongly suggested. Please refer to the course notebooks[1] for a sketch of the analysis to be performed.

Network analysis must include at least:

- Degree distribution analysis;
- Connected components analysis;
- **2.3** Path analysis;
- Clustering Coefficient, Density analysis;
- Centrality analysis.

Moreover, the statistics computed on the crawled data must be compared with the ones of (i) ER and (ii) BA graphs having (almost) *the same number of nodes and edges.*

For simplicity's sake perform the analysis on the simple (i.e., static non-higher-order) undirected, unweighted version of the built network.

[1]`https://github.com/sna-unipi/SNA_lectures_notebooks`

Part 3: Network Manipulation and Analysis

Each group **must** address *at least N* among the following tasks (where N is the number of group components).

> ☞ **Note:** Tasks are grouped into 2 clusters: (i) analytical and (ii) network manipulation. When $N > 1$ the selected tasks must be equally selected among the two clusters.
>
> For even values of N, the group is free to select where to allocate the majority of the tasks (e.g., N=3: 1 task in analysis, 2 in manipulation - or vice versa).

Cluster 1: Data analysis

Community Detection: Identify, evaluate and validate the modular structure of the crawled network sample. The results of at least 3 CD algorithms of your choice (e.g., K-clique, Label Propagation, Louvain, Infomap, Demon/Angel...) must be evaluated and compared. If additional semantic information for the analyzed graph is available use them to make sense of the identified partitions. For CD algorithm implementations (as well as for their evaluation and comparison) refer to the CDlib (https://github.com/GiulioRossetti/cdlib) library.

Dynamic Community Detection: Generate, starting from your dataset, a series of snapshots. To do so - in case of unavailability of temporal edge/node annotation - for each snapshot (independently) randomly select 30% of the edges from the original network. After that: (i) partition the obtained dynamic network in communities implementing a two-step approach as discussed during the course (using a static CD algorithm of your choice) (ii) provide a simple definition for the main community events (e.g., merge, split, growth, shrink), and (iii) analyze them. In case the crawled data contain temporal information you can avoid the random sampling and apply a temporal discretization of the original network.

Spreading: Simulate, using the NDlib (https://github.com/GiulioRossetti/ndlib) python library, the diffusion models discussed during the course (i.e., SI, SIS, SIR, and Threshold model) both on the crawled data and on synthetic graphs (i.e., ER and BA). Analyze the simulation results varying both model parameters and initial conditions (i.e., the infection seeds).

Opinion Dynamics: Simulate, using the NDlib python library, the opinion dynamics models discussed during the course (i.e., Voter, Snayzd, Majority Rule, Q-Voter, Deffuant w/o bias) both on the crawled data and in mean-field settings (i.e., complete graph). Analyze the simulation results varying both model parameters and initial conditions.

Link Prediction: Partition the network in a training (80% of the edges) and a test set (20% of the edges) and apply some of the classical unsupervised link prediction approaches introduced in "David Liben-Nowell, Jon M. Kleinberg: The link prediction problem for social networks. CIKM 2003" (i.e. Common Neighbors, Adamic Adar, Jaccard, Preferential

Attachment). Discuss the prediction accuracy as done in the referenced paper.

Link Prediction 2: Following the same rationale of the previous exercise, design a supervised approach[2] to link prediction using a classifier. Define the features, test the model(s), evaluate and discuss the obtained results.

Network Resilience: Define a set of measures to compute tie strength and analyze the impact of strong/weak ties on the connectedness and resilience of the crawled network.

Cluster 2: Network Manipulation

Graphlets: Graphlets are small, connected, non-isomorphic[3] induced subgraphs[4] of a large network. The size of a graphlet is the number of the nodes it is composed of: for the same size multiple graphlets may exist[5]. Define an approximate algorithm that allows you to estimate the number of graphlets of sizes 3 and 4 and test it on your data. Which are the most frequent graphlets?

Community Detection 2: Define, implement and test either an existing or a novel Community Detection approach not yet present in CDlib. For algorithmic ideas and/or a list of well-known approaches refer to the Fortunato and Coscia's surveys (or contact the course instructors).

Spreading 2: Leveraging the Custom Model facility offered by NDlib design an ad-hoc, novel, diffusion model for the crawled graph. The model can be designed to take advantage of both network topological characteristics as well as external semantic information attached to nodes/edges (if present). Define your model so as to solve a specific diffusion problem you consider interesting for your data. The model can be either coded in Python or expressed using NDQL. Analyze the results varying both model parameters and initial conditions (i.e., the infection seeds).

Opinion Dynamics 2: Design an ad-hoc, novel, opinion dynamics model for the crawled graph that extends the Deffuant one (e.g., modeling the effect of social media, stubborn actors, or peer pressure). The model can be designed to take advantage of both network topological characteristics as well as external semantic information attached to nodes/edges (if present). Define your model so as to solve a specific diffusion problem you consider interesting for your data. Analyze the results varying both model parameters and initial conditions.

[2]This exercise requires knowledge of Data Mining tools and techniques.

[3]Graph isomorphism is an equivalence relation on graphs: two graphs G and H are said to be isomorphic if there exist a function f such that any two vertices u and v of G are adjacent in G if and only if $f(u)$ and $f(v)$ are adjacent in H. This kind of bijection is commonly described as "edge-preserving bijection".

[4]An induced subgraph must contain all edges between its nodes that are present in the original network.

[5]A single graphlet exists only among 2 nodes. Considering all the possible ways to connect 3 nodes, two different graphlets can be identified: the chain, and the triangle.

Feature-rich Network Analysis: Analyze the interplay of node/edge attributes w.r.t. the modeled topology. Is it possible to observe relevant multiscale mixing patterns (i.e., using Conformity (https://github.com/GiulioRossetti/conformity) or other node-related measures (https://github.com/piratepeel/MultiscaleMixing) on subpopulations)? Do exist correlations, at the node level, between some topological attribute (e.g., clustering, centrality...) and the entropy/purity of the node's neighborhood attributes?

Dynamic Network Analysis: Using DyNetX (or similar libraries) create a Stream Graph/Network Snapshot representation of the collected data describing a dynamic phenomena. Analyze the built temporal graph and study its stability.

Higher-order Network Analysis: Using halp (https://github.com/Murali-group/halp) or HyperNetX (https://github.com/pnnl/HyperNetX) create a Hypergraph representation of the collected data describing higher-order phenomena. Analyze the built topology focusing on the properties of its hyperedges.

Link Prediction 3: Assuming a dynamic network (modeled as a snapshot sequence) implement a (simple) pipeline for predicting new links leveraging the historical information available. Validate the pipeline on your data focusing on the prediction of intra-community interactions. Use as a reference example the methodology discussed during class as a case study.

Part 4: Open Question

Define a research question on your data and use SNA tools to address it!

This task requires you to:

1. reason on the crawled data,
2. identify a concrete research question, and
3. (try to) address it by *combining* the technique discussed in class and/or *implementing* your own ideas/solutions.

Students are encouraged to leverage, along with network topology, the additional information (e.g., nodes' attributes, edge weights, temporal information. . .) collected during the crawling stage. Mixing analytical methodologies acquired during your learning path with SNA ones is considered a plus.

☞ **Warning:** Considering that the tasks proposed in part 3 cover standard problems for which technical solutions are (at least) partially provided by the course's notebooks, the open questions will have an **higher weight** on the project evaluation.

☞ **Research questions examples:**

- How does Covid19 affected SoundCloud users' listenings?
- Which are the pull/push factors that affects mobility of highly skilled individuals (e.g., professionals on LinkdIn, researchers on MAG)?
- Is it possible to identify bots by the patterns of their online activities?
- What is an echo-chamber and how can we track it on Twitter?
- ...

Part 5: Project Report

Discuss the result of all the analysis (parts 0 to 4) in a written report:

1. Specify group members and link the GitHub repositories on the first page;
2. Focus on the analytical methodologies applied and obtained results;
3. Max 15 pages, double column (*use the template provided in the repository report folder*).

☞ **Warning:** Notice that a *mere list of the analytical results* not supported by an interpretation tied to the collected data **will not be accepted** as a valid contribution.

Example:

- When discussing and comparing community structures try to analyze the partition quality as a factor of external data (e.g., studying the homophily of nodes' labels within the same cluster;
- While comparing the results obtained on your data with reference ones comment on the similarities/divergences;
- ...