ADVANCED LEARNING FOR TEXT AND GRAPH DATA
MASTER DATA SCIENCE
Lab 1: Identification of Influential Entities in Spreading Processes

Maria Rossi, Antoine Tixier and Michalis Vazirgiannis

January 26, 2017

1 Description

The goal of this lab is to study several techniques related to identifying influential spreaders in complex networks and to examine spreading processes on real data. We give a brief introduction of the basic techniques that will be used in this lab, and then we describe the tasks that need to be performed.

2 Identifying Influential Nodes

Spreading processes have been gaining great interest in the research community. This is justified mainly by the fact that they occur in a plethora of applications ranging from the spread of news and ideas to the diffusion of influence and social movements and from the outbreak of a disease to the promotion of commercial products. A fundamental aspect of understanding and controlling the spreading dynamics is the identification of influential spreaders that can diffuse information to a large portion of the network. The problem of identifying nodes with good spreading properties in networks, can be split in two subtopics: (i) identification of individual influential nodes and (ii) identification of a group of nodes that, by acting all together, are able to maximize the total spread of influence.

Let a graph $G = (V, E)$ represent a social network consisting of $|V|$ users represented as nodes and $|E|$ edges representing the relationships between users. To identify the individual influential spreaders, one needs to find the metric (see section 2.1.2) that succeeds in correctly ranking all the nodes according to their spreading capabilities. The spreading capabilities of each node is calculated after triggering a diffusion process in the graph from the node in question. The diffusion is simulated via a chosen spreading model (see section 2.1.1).

For the latter case, which is also called the *Influence Maximization (IM)* problem, the problem can be formulated as follows: given a social network, a diffusion model with some parameters and a number *l*, find a seed set S $\subset$ V of at most *l* nodes such that the influence spread is maximized.
Next we will briefly present some of the most used models that simulate a diffusion process.

2.1 Identifying Single Spreaders

2.1.1 Modelling a Spreading Process

Epidemic models

The models that are most commonly used in order to simulate a spreading process, specifically for the case of the identification of individual influential spreaders, are borrowed from the field of epidemics [1, 2]. Here we will specifically get involved with the *Susceptible-Infected-Revovered (SIR)* model. The model assumes a population of N individuals, divided into the following three states. Susceptible (S): the individual is not yet infected, thus being susceptible to the epidemic; Infected (I): the individual has been infected with the disease and is capable of spreading the disease to the susceptible population; Recovered (R): after an individual has experienced the infectious period, it is considered as immune to the disease and it is not able to be infected again or to transmit the disease to others. Every node that is on the I state can infect its susceptible neighbors with probability β (called infection rate) and afterwards it can recover with probability γ (called recovery rate). The following state diagram represents the behavior of each node of the network:

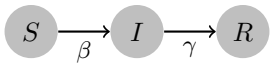

Figure 1: SIR model.

Initially, all the nodes of the network are set at the susceptible state S, except the one whose spreading properties are to be studied, which is set at the infected state I. Then, we let the process evolve as described above. Note that at a given timestep a node can only recover after he was given a chance to infect its neighbors.

In our approach, we set β close to the epidemic threshold $\tau = 1/\lambda_1$, where λ_1 is the largest eigenvalue of the adjacency matrix of the network [3] . We also set parameter γ= 0.8, as used by Kitsak et al. [4].

2.1.2 Metrics that Identify Influential Nodes

There are numerous topological metrics that have been used in order to locate the privileged nodes that will trigger an effective information diffusion. We present you here three of the most common ones which we will be also using in this lab.

Degree Centrality

Degree centrality is the simplest index to identify nodes influences: the more connections a node has, the greater the influence of the node gets. Obviously d_{v_i} equals the number of node v_i's neighbors. We denote as D the set of top k_{max} degree nodes (same cardinality as set C for comparison purposes).

Core number Centrality [5]

Given our undirected simple network G, C_k is defined to be the k-core subgraph of G if it is a maximal connected subgraph in which all nodes have degree at least k. Then, each node $v \in V$ has a core number $c_{v_i} = k$, if it belongs to a k-core but not to a $(k+1)$-core. Let us denote as C the set of nodes with the maximum core number *kmax*.

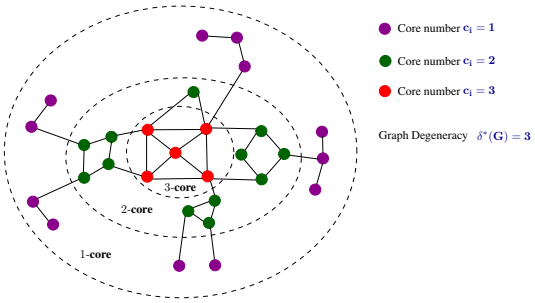

Figure 2: *k-core* decomposition illustration.

Truss number Centrality [6, 7]

The K-truss decomposition extends the notion of k-core using triangles, i.e., cycle subgraphs of length 3. The K-truss subgraph of G, denoted by T_K, K $\geq$ 2, is defined as the largest subgraph where all edges belong to K-2 triangles. Respectively, an edge $e \in E$ has truss number $t_{e_i} = K$ if it belongs to T_K but not to $T_K + 1$. Since the definition of K-truss is per edge, we define the nodes truss number t_{v_i}, $v_i \in V$ as the maximum t_{e_i} of its adjacent edges. Let us denote as T the set of nodes with maximum node truss number (in other words, this set contains the nodes of the maximal K-truss subgraph).

It has been shown that the maximal k-core and K-truss subgraphs (i.e., maximum values for k;K) overlap, with the latter being a subgraph of the former; in fact, K-truss represents the core of a k-core that fillters out less important information. Fig. 3 shows an example of a graph and its k-core and K-truss subgraphs respectively.

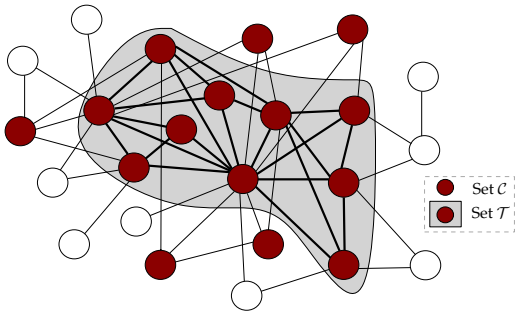

Figure 3: Schematic representation of the maximal k-core and K-truss subgraphs of a graph.

2.1.3 Tasks to be Done in the Lab

Under each directory *SingleSpreaders/SIR/scenario*/group** you will find the results of the SIR model applied on the $WikiVote$ dataset (*https://snap.stanford.edu/data/wiki-Vote.html*) which can be found as an edgelist under *data/undirected_gcc*. The network contains all the Wikipedia voting data from its inception to 2008. Nodes represent users, and a directed edge from node i to node j represents that user i voted for user j. We provide the undirected and greatest connected component version of the graph for which $|V| = 7,066$, $|E| = 100,736$, $k_{max} = 53$ and $K_{max} = 23$. (Here we use the undirected form of the graph as the k-core and K-truss decomposition algorithms are defined for undirected graphs.)

Let us note that, because of the probabilistic nature of the SIR model, when it is applied to the same node, the results can differ from run to run. Not only the number of nodes being infected can differ, but also the duration of the process. In order to have the average behavior of the node, one needs to repeat the process multiple times for the same nodes and average the results.

The aforementioned directories contain *.txt* files that contain the average behavior of every node (the id of whom appears as the title of the file). In each file the first column represents the timestep of the process, the second column represents the number of nodes that passed to the infected state I in the specific timestep, the third column represents the cumulative number of nodes that passed to the infected state I until the specific timestep and the fourth column represents the percentage of the nodes of the network that passed to the infected state until the specific timestep.

scenario1/groupA: results after triggering the SIR model from nodes that belong to set D. Parameters used: $\beta = 0.01$, $\gamma = 0.8$.

scenario1/groupB: results after triggering the SIR model from nodes that belong to set C $\setminus$ T (reminder: T is a subset of C and here we are interested in the nodes that only belong to the k-core subgraph of the graph). Parameters used: $\beta = 0.01$, $\gamma = 0.8$.

scenario1/groupC: results after triggering the SIR model from nodes that belong to set T. Parameters used: $\beta = 0.01$, $\gamma = 0.8$.

scenario2/groupC: results after triggering the SIR model from nodes that belong to set T. Parameters used: $\beta = 0.03$, $\gamma = 0.8$.

scenario3/groupC: results after triggering the SIR model from nodes that belong to set T. Parameters used: $\beta = 0.05$, $\gamma = 0.8$.

- Print in a table the average number of nodes that are being infected at every timestep (2nd column) until the 10th timestep for each group in *scenario1*. Then print the cumulative number of nodes being infected (3rd column) for all the timesteps of the process. What do you observe?
  - *Tip: Remember that the process triggered by every node can last a different number of timesteps.*
- Plot the average number of nodes that are being infected at every timestep (2nd column) for *groupC* of nodes for every different scenario. What do you observe?

2.2 Identifying Multiple Spreaders

In the case of the Influence Maximization problem, we are searching for the l nodes that will maximize the influence spread in our network while the spreading process is simulated by a diffusion model.

2.2.1 Modeling a Spreading Process

In their seminal work Kempe et. al [8] formulated the problem of *Influence Maximization* while adopting two diffusion models borrowed from mathematical sociology: the *Linear Threshold (LT)* and the *Independent Cascade (IC)* model. Both are until today broadly used in order to simulate diffusion processes in order to attack the *IM* problem. Below, we briefly describe the *Linear Threshold* model, which we will be using in the remainder of this exercise.

The Linear Threshold (LT) model

In this model, a node v_i is influenced by its neighbor v_j according to a weight b_{ij}. The sum of all the weights towards all neighbors of v_i must be less than or equal to 1. The weights can either i) equal to $1/d(v_i)$ or ii) equal to the real influence that a user v_j can have to v_i which is an ongoing research topic. Each node v_i chooses a threshold θ_{v_i} uniformly at random from the interval [0,1]. It represents the weighted fraction of v_is neighbors that must become active in order for v_i to become active as shown in equation 1. Given a random choice of thresholds and an initial set of active nodes (with all other nodes being inactive), the diffusion process unfolds in discrete steps.

$$\sum_{v_j \; active \; neighbor \; of \; v_i} b_{ij} \geq \theta_{v_i} \; (1)$$

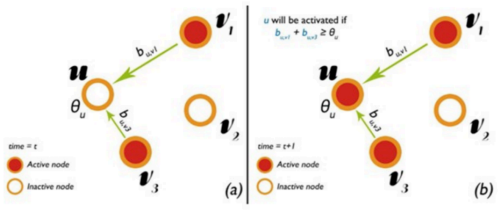

Figure 4: Linear Threshold model illustration.

2.2.2 Algorithms for the IM problem

The Greedy Algorithm

Kempe et. al [8] proved that the function of the influence spread under both *LT* and *IC* models is monotone non-decreasing and submodular. A monotone non-decreasing submodular function can be maximized by a greedy algorithm [9]. By exploiting these properties, they presented a greedy algorithm that achieves (1-1/e) approximation ratio.

Quick reminder: *What is a submodular function?*
If V is a finite set, a submodular function is a function f: $2^V \rightarrow \mathbb{R}$ which satisfies the following:

- $\forall$ S $\subseteq$ T $\subseteq$ V $\setminus$ v we have that: f (S $\cup$ v) - f(S) $\geq$ f (T $\cup$ v) - f(T) (**property of diminishing returns**)

f is monotone non-decreasing if and only if:

- for all S $\subseteq$ T $\subseteq$ V, f(S) $\leq$ f(T)

Algorithm 1 The Greedy Algorithm

1: Initialize $S = \emptyset$
2: **for** i=1 to l **do**
3: select
4: $u = \arg\max_{w \subset V \setminus S}[f(S \cup w) - f(S)]$
5: $S = S \cup u$
6: **end for**
7: return S

Where **f** is the function that calculates the influence spread of a node or a set of nodes. In this case, to calculate the influence spread we simulate a spreading process via the LT model starting from the nodes of interest - we perform Monte Carlo simulations. The total number of nodes being activated at the end of the process is the value needed. Again here, as the model is probalistic, multiple Monte Carlo simulations are performed in order to calculate the output value of function f which is the total number of nodes being active at the end of a process on average. l is the number of nodes for which the influence is maximized which is chosen by the user.

2.2.3 Tasks to be Done in the Lab

Under each directory *MultipleSpreaders/Greedy/** you will find the results of the Greedy Algorithm for IM applied on the $WikiVote$ dataset (found under *data/directed*) which is now used in its directed format. Here we use the directed version of the network as the Linear Threshold model that is used to simulate the spreading is defined for directed graphs. Every node v_i is influenced by node v_j according to a weight b_{ij} that we set equal to $1/d(v_i)$ (where $d(v_i)$ is the degree of v_i).

In each file, the first column contains the IDs of the nodes that have been iteratively selected by the greedy algorithm. The second column represents the cumulative influence (average number of nodes activated) having triggered the diffusion from the nodes up to the current row. The sixth column represents computation time. In this exercise we won't be interested in the data of the rest of the columns.

Greedy/MC_100 contains the results after running the Greedy Algorithm while performing 100 Monte Carlo simulations for every calculation of the influence spread.

Greedy/MC_10000 contains the results after running the Greedy Algorithm while performing 10000 Monte Carlo simulations for every calculation of the influence spread.

- Plot the influence achieved by the $l = 1$ to 30 nodes identified by the Greedy Algorithm in the two cases. Compare the influence achieved while having run the Greedy Algorithm using 100 and 10.000 Monte Carlo simulations. Compare the running time of the different algorithms. What is the complexity of the Greedy Algorithm? Are there any similarities in the nodes identified in the two cases?

- Compare the degree and core number centrality of the nodes identified in the MC_10000 case. What do you observe? How are those centralities compared to the highest degree and the highest core number ? (Use the undirected format of the network to calculate the degree and core number of those nodes.)

References

[1] William O. Kermack, and Anderson G. McKendrick. "A contribution to the mathematical theory of epidemics." Proceedings of the Royal Society of London A: mathematical, physical and engineering sciences. Vol. 115. No. 772. The Royal Society, 1927.

[2] Mark EJ. Newman. "Spread of epidemic disease on networks." Physical review E 66.1 (2002): 016128.

[3] Deepayan, C., Yang, W., Chenxi, W., Jurij, L. & Christos, F. Epidemic thresholds in real networks. ACM Trans. Inf. Syst. Secur. 10(4), 1:11:26 (2008)

[4] Maksim Kitsak, Lazaros K. Gallos, Shlomo Havlin, Fredrik Liljeros, Lev Muchnik, H. Eugene Stanley, and Hernn A. Makse. "Identification of influential spreaders in complex networks." Nature physics 6.11 (2010): 888-893.

[5] Vladimir Batagelj and Matjaz Zaversnik. "An O (m) algorithm for cores decomposition of networks." arXiv preprint cs/0310049 (2003).

[6] Jia Wang and James Cheng. "Truss decomposition in massive networks." Proceedings of the VLDB Endowment 5.9 (2012): 812-823.

[7] Fragkiskos D. Malliaros, Maria-Evgenia G. Rossi, and Michalis Vazirgiannis. "Locating influential nodes in complex networks." Scientific reports 6 (2016).

[8] David Kempe, Jon Kleinberg, and va Tardos. "Maximizing the spread of influence through a social network." Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2003.

[9] Nemhauser, George L., Laurence A. Wolsey, and Marshall L. Fisher. "An analysis of approximations for maximizing submodular set functionsI." Mathematical Programming 14.1 (1978): 265-294.