# Analiza socijalnih mreža

# NetworkX biblioteka za analizu socijalnih mreža

Marko Mišić, Jelica Protić

13M111ASM

2019/2020.

# Uvod (1)

- NetworkX je Python biblioteka za stvaranja, manipulaciju i proučavanje strukture, dinamike i funkcija kompleksnih mreža
- Biblioteka obezbeđuje:
  - Strukture podataka za neusmerene i usmerene grafove i multigrafove
  - Veliki broj standardnih algoritama za rad sa grafovima
  - Otkrivanje strukture grafa i izračunavanje mrežnih metrika
  - Generatore za klasične i *random* mreže
  - Efikasnu manipulaciju atributima mreže
- Može se preuzeti na:
  - https://networkx.github.io/
  - Aktuelna verzija 2.4

# Uvod (2)

- Dodatno, biblioteka obezbeđuje:
  - Alate za proučavanje strukture i dinamike socijalnih, bioloških i infrastrukturnih mreža
  - Standardan programski interfejs i implementaciju grafa pogodnu za mnoge primene
  - Interfejs ka postojećim implementacijama numeričkih algoritama u C, C++ i Fortran
  - Skalabilan rad sa velikim skupovima podataka
  - Vizuelizaciju mreže
- NetworkX podržava rad sa velikim brojem standardnih grafovskih formata

# Instalacija

- Biblioteku je potrebno instalirati da bi se koristila
  - Korišćenjem Python **`pip`** menadžera paketa
  - U okviru Python distribucije kao što je Anaconda
- Korišćenjem paket menadžera:
  - **`pip install networkx`**
- Korišćenjem Anaconda okruženja:
  - Pokrenuti Anaconda Prompt
  - **`conda install -c anaconda network`**
- Radi proširenja mogućnosti NetworkX-a, preporučuje se instalacija i sledećih paketa:
  - Matplotlib, pandas, NumPy, SciPy, PyGraphviz, pydot, lxml
  - **`pip install networkx[all]`**
  - **`pip install numpy scipy pandas matplotlib pygraphviz pydot pyyaml gdal`**

# Stvaranje grafa (1)

- Da bi se radilo sa NetworkX bibliotekom, najpre je potrebno uključiti je u Python program:
  - `import networkx as nx`
- Po definiciji, graf je kolekcija čvorova i grana
  - U NetworkX, čvorovi mogu biti bilo kakvi objekti koji se mogu heširati (*hashable*)
    - String, slika, XML dokument, drugi graf i sl.
- Stvaranje praznog objekta grafa:
  - `G = nx.Graph()`
  - Moguće dodavanje inicijalizatora za punjenje grafa

# Stvaranje grafa (2)

- Postoje različiti tipovi grafovskih objekata:
  - Graph – neusmereni graf sa petljama
    - `g = nx.Graph()`
  - DiGraph – usmereni graf sa petljama
    - `d = nx.DiGraph ()`
  - MultiGraph – neusmereni graf sa paralelnim granama
    - `m = nx.MultiGraph ()`
  - MultiDiGraph – usmereni graf sa paralelnim granama
    - `h = nx.MultiDiGraph ()`
- Postoje rutine za konvertovanje:
  - U neusmereni graf: `g.toundirected()`
  - U usmereni graf: `g.todirected()`

# Stvaranje grafa (3)

- Čvorovi se mogu dodavati na nekoliko načina:
  - Dodavanjem jednog po jednog čvora

    ```
    G.add_node(1)
    ```
  - Dodavanjem liste čvorova

    ```
    G.add_nodes_from([2, 3])
    ```
  - Dodavanjem čvorova nekog drugog grafa

    ```
    H = nx.path_graph(10)
    G.add_nodes_from(H)
    ```

    Pravi linijski graf sa 10 čvorova
- Čvorovima se mogu pridružiti i atributi
  - Čvor je tada potrebno dodati kao torku koja sadrži identifikator čvora i rečnik koji sadrži atribute

# Stvaranje grafa (4)

- Grane se mogu dodavati na nekoliko načina:
  - Dodavanjem jedne po jedne grane

```
G.add_edge(1, 2)
e = (2, 3)
G.add_edge(*e)
```

  - Dodavanjem liste grana

```
G.add_edges_from([(1, 2), (1, 3)])
```

  - Dodavanjem *ebunch* objekta koji sadrži grane
  - Objekat *ebunch* predstavlja iterabilni kontejner torki grana

```
G.add_edges_from(H.edges)
```

- Torke grana mogu biti u formi:
  - 2-torke koja sadrži čvorove koje grana spaja
  - 3-torke koja sadrži čvorove i rečnik sa atributima grane
    - `(2, 3, {'weight': 3.1415})`

# Stvaranje grafa – primer (1)

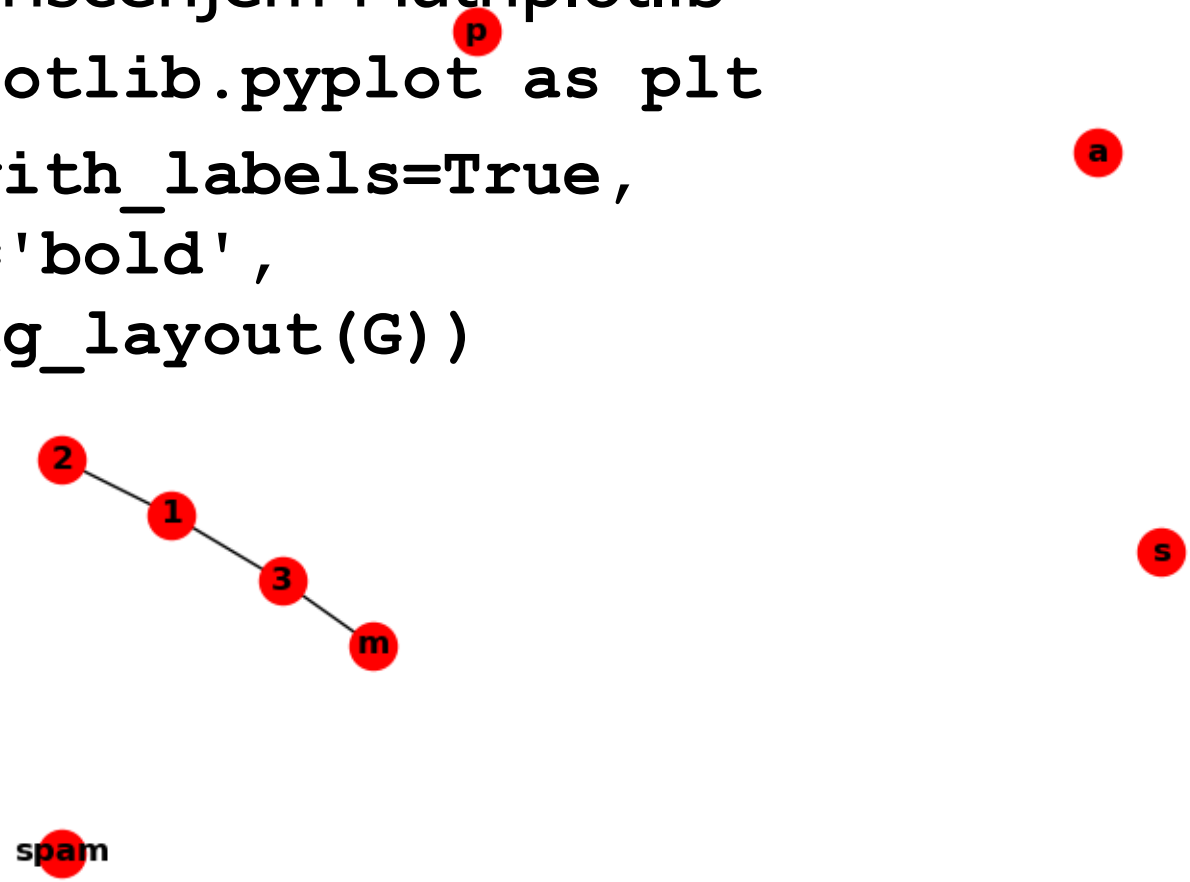
- Primer proizvoljnog grafa:

```
G.add_edges_from([(1, 2), (1, 3)])
G.add_node(1)
G.add_edge(1, 2)
G.add_node("spam")
    # adds node "spam"
G.add_nodes_from("spam")
    # adds 4 nodes: 's', 'p', 'a', 'm'
G.add_edge(3, 'm')
```

# Stvaranje grafa – primer (2)

- Iscrtan graf korišćenjem Mathplotlib

```
import matplotlib.pyplot as plt
nx.draw(G, with_labels=True,
font_weight='bold',
pos=nx.spring_layout(G))
plt.show()
```



# Ispitivanje grafa (1)

- Dohvatanje broja čvorova
  - **`G.number_of_nodes()`**
- Dohvatanje broja grana
  - **`G.number_of_edges()`**
- Dohvatanje pogleda na elemente grafa
  - Ne omogućavaju promenu (*read only view*)
  - **`G.nodes()`** – skup čvorova grafa
  - **`G.edges()`** – lista torki koje predstavljaju grane
  - **`G.adj()`** – rečnik koji predstavlja listu susednosti svakog čvora zajedno sa atributima grana
  - **`G.degree()`** – rečnik koji sadrži parove **`čvor:stepen`**
  - Pogledi se po potrebi mogu konvertovati u druge objekte

# Ispitivanje grafa (2)

- Pogled se može napraviti samo na podskup podataka
  - Podskup se zadaje se kao opcioni parametar u obliku *nbunch* objekta
  - Objekat *nbunch* može biti:
    - **`None`** – pogled na sve čvorove
    - Pojedinačni čvor
    - Iterabilni kontejner koji sadrži čvorove, a da pritom nije sam čvor grafa
- Kroz poglede zasnovane na rečniku se može iterirati
  - Korišćenjem **`.items()`** ili **`.data()`** funkcija
- Dohvatanje suseda čvora:
  - **`g.neighbors ()`**

# Ispitivanje grafa (3)

- Pristup grana grafa je moguć i kroz upotrebu operatora []
  - `G[1]` – pristupa skupu grana zadatog čvora
    - `AtlasView({2: {}, 3: {}})`
  - `G[1][2]` – pristupa atributima grane između čvorova 1 i 2
- Na ovaj način se mogu dodavati i atributi postojećim granama
  - `G.add_edge(1, 3)`
  - `G[1][3]['color'] = "blue"`
  - `G.edges[1, 2]['color'] = "red`
  - `G[1]`
    - `AtlasView({2: {'color': 'red'}, 3: {'color': 'blue'}})`

# Uklanjanje čvorova iz grafa

- Pražnjenje objekta grafa:
  - `G.clear()`
- Uklanjanje čvorova:
  - `G.remove_node(2)`
  - `G.remove_nodes_from("spam")`
- Uklanjanje grana:
  - `G.remove_edge(1, 3)`
  - `G.remove_edges_from([(1, 3), (1,2)]`

# Atributi grafa (1)

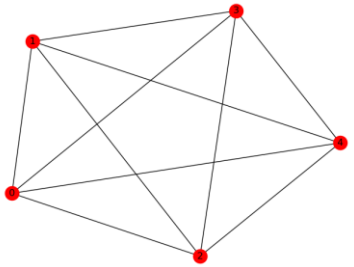
- Atributi se mogu dodeliti na nivou celog grafa, pojedinačnih čvorova ili grana
- Atributi grafa:
  - `G = nx.Graph(day="Friday")` ili `G.graph['day'] = "Monday"`
  - `G.graph`
    - `{'day': 'Friday'}`
- Atributi čvora:
  - Čvorovi se prethodno moraju dodati
  - `G.add_node(1, time='5pm')`
  - `G.add_nodes_from([3], time='2pm')`
  - `G.nodes[1]`
    - `{'time': '5pm'}`
  - `G.nodes[1]['room'] = 714`
  - `G.nodes.data()`
    - `NodeDataView({1: {'time': '5pm', 'room': 714}, 3: {'time': '2pm'}})`

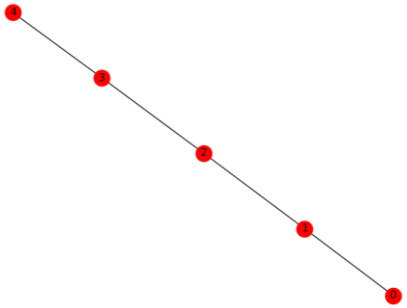
# Atributi grafa (2)

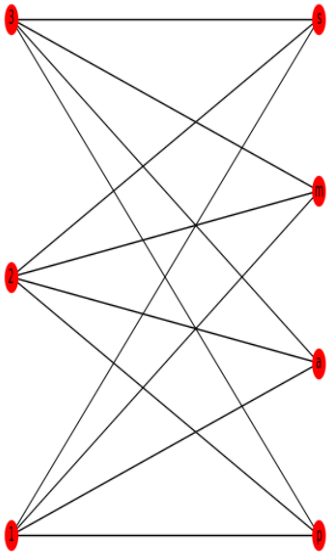
- Atributi grane:
  - Grane se najpre moraju dodati
  - `G.add_edge(1, 2, weight=4.7 )`
  - `G.add_edges_from([(3, 4), (4, 5)], color='red')`
  - `G.add_edges_from([(1, 2, {'color': 'blue'}), (2, 3, {'weight': 8})])`
  - `G[1][2]['weight'] = 4.7`
  - `G.edges[3, 4]['weight'] = 4.2`
- Specijalni atributi grane je **`weight`**
  - Mora biti numeričkog tipa

# Generatori grafa (1)

- NetworkX obezbeđuje određeni broj generatora mreža
- Jednostavni generatori:
  - Kompletan graf
    - **`nx.complete_graph(5)`**
  - Linijski graf
    - **`nx.path_graph(5)`**
  - Bipartitni graf
    - **`nx.complete_bipartite_graph(n1 , n2)`**
  - Pravilna mreža proizvoljnih dimenzija (*lattice*)
    - **`nx.grid_graph ([10 ,10 ,10 ,10])`**
      - # 4D, 100^4 čvorova







# Generatori grafa (2)

- Generisanje *random* grafova:
  - Erdos-Renyi *random* graf
    - Sa zadatom verovatnoćom **`p`**:

      **`nx.gnp_random_graph(n,p)`**
    - Sa zadatim brojem grana **`m`**:

      **`nx.gnm_random_graph(n, m)`**
  - Preferencijalno vezivanje

    **`nx.barabasi_albert_graph(n, m)`**
  - Watts-Strogatz model

    **`nx.watts_strogatz_graph(n, k, p)`**

# Čitanje i pisanje grafova

- NetworkX podržava veliki broj grafovskih formata
  - Liste susednosti i liste grana
    - Opis formata u NetworkX dokumentaciji
  - GML, Pickle, GraphML, YAML, Pajek, GEXF, LEDA, SparseGraph6, GIS Shapefile, JSON
  - Čitanje CSV datoteka kroz **`pandas`** biblioteku
- Primer – čitanje i pisanje GML formata:
  - **`nx.write_gml(red, "path.to.file")`**
  - **`mygraph = nx.read_gml("path.to.file")`**

# Mere centralnosti

- Centralnost po stepenu:
  - `degree_centrality(G)`
  - `in_degree_centrality(G)`
  - `out_degree_centrality(G)`
- Centralnost po svojstvenom vektoru:
  - `eigenvector_centrality(G)`
- Centralnost po bliskosti:
  - `closeness_centrality(G)`
- Relaciona centralnost:
  - `betweenness_centrality(G)`
- Sve metrike se mogu dodatno parametrizovati
  - Normalizacija, upotreba drugačijeg načina računanja, itd.

# Svojstva grafa

- Računanje najkraćeg puta između parova čvorova **`s`** i **`t`**:
  - **`nx.shortest_path(G,s,t)`**
- Određivanje dijametra grafa:
  - **`nx.diameter(G)`**
- Određivanje ekscentričnosti čvora:
  - **`eccentricity(G)`**
- Računanje prosečnog koeficijenta klasterizacije:
  - **`nx.average_clustering(G)`**

# Detekcija komuna

- Potrebno je dodatno uključiti paket:
  - `from networkx.algorithms import community`
- Detekcija komuna bazirana na modularnosti:
  - *Clauset-Newman-Moore* pohlepni algoritam
  - `greedy_modularity_communities(G)`
- Girvan-Newman-ov algoritam:
  - `girvan_newman(G)`
- Propagacija labela:
  - `label_propagation_communities(G)`
  - `asyn_lpa_communities(G)`

# Primena u socijalnim mrežama

- Gotovi generatori mreža:
  - Zachary's Karate Club
    - **`karate_club_graph()`**
  - Les Miserables
    - **`les_miserables_graph()`**
  - Generisanje *scale-free* mreže:
    - **`scale_free_graph(n[, alpha, beta, gamma, …])`**
- Provera da li mreža prati *power law*:
  - Nije implementirana direktno
  - Odrediti distribuciju čvorova po stepenu
  - Iskoristiti paket kao što je **`powerlaw`** za fitovanje krive

# Iscrtavanje grafova

- Kroz Mathplotlib, Graphviz i pydot biblioteke
  - **`nx.draw(G, [pos])`** metoda
  - Opcioni parametar **`pos`** se koristi za određivanje pozicija čvorova korišćenjem nekog rasporeda
- Podržan je veliki broj rasporeda:
  - **`random_layout(G)`** – slučajne pozicije čvorova
  - **`shell_layout(G)`** – raspored u koncentričnim krugovima
  - **`spring_layout(G)`** – baziran na Fruchterman-Reingold
  - **`spiral_layout(G)`** – spiralni raspored
  - **`bipartite_layout(G, n1)`** – bipartitini graf
- Pozvati **`show()`** iz Mathplotlib da bi se graf prikazao

# Literatura


- NetworkX, https://networkx.github.io/
- NetworkX tutorial, https://networkx.github.io/documentation/stable/tutorial.html
- NetworkX installation guide, https://networkx.github.io/documentation/stable/install.html
- Jacob Bank, NetworkX tutorial, Stanford SNAP, 2012. http://snap.stanford.edu/class/cs224w-2012/nx_tutorial.pdf