Università degli Studi di Udine

Corso di Internet of Things, Big Data and Web

Primo progetto di Social Computing

CREAZIONE ED ANALISI DI UNA RETE SOCIALE

Allievi:
Rizzetto Sasha [142660]
Gasparollo Denis [143225]
Bensi Martino [142793]
Francescut Matteo [143352]

Anno Accademico 2020-21

Indice

1 Introduzione

Lo scopo del progetto è la realizzazione e lo studio di una **rete sociale**. A tal fine i dati necessari saranno ottenuti mediante le API di Twitter.

1.1 Rappresentazione della rete

La rete risultante sarà rappresentata da un grafo **così strutturato**:

- Ogni *nodo* sarà caratterizzato dall'id del relativo utente
- Ogni *arco* rappresenterà una relazione di "follow" tra due utenti
- Gli *attributi* di ogni nodo conterranno i dettagli del relativo utente
- Il *numero di follower* sarà riportato come attributo di ogni nodo

Il grafo sarà rappresentato mediante una visualizzazione interattiva, sfruttando la libreria Python *pyvis*. Tale visualizzazione sarà fruibile mediante una pagina HTML.

1.2 Analisi condotte

Sul grafo verranno effettuate **analisi** in merito a verifiche di *connessione* e di *bipartizione*, misurazioni di *distanze*, misurazioni di *centralità* e relative correlazioni tramite coefficienti di Pearson e Kendall, calcolo della *copertura minima* degli archi, osservazione del fenomeno *"small-worldness"* con coefficienti Sigma e Omega.

1.3 Osservazione delle "cricche"

Infine sarà costruito un *sottografo* del grafo iniziale a partire dall'utente *KevinRoitero* (scelto per motivi di complessità computazionale). Lo scopo di ciò sarà osservare le *cricche*, calcolando la cricca massima e la sua dimensione.

1.4 Repository GitHub

I codici sorgenti del progetto, nonchè i dati scaricati e le rappresentazioni prodotte, sono reperibili sulla seguente repository GitHub: `https://github.com/sasharizzetto/social_computing`

2 Strumenti impiegati

Il linguaggio scelto per lo svolgimento del progetto è *Python*. L'intero codice sorgente del progetto è stato organizzato in un Jupyter Notebook, diviso punto per punto con una descrizione per ogni parte. Per questa ragione in questa relazione saranno riportati solamente i frammenti indispensabili di codice. Le principali *librerie* impiegate sono:

- **Tweepy**: semplifica l'accesso e l'utilizzo delle API di Twitter
- **NetworkX**: permette la creazione, manipolazione e l'analisi di reti
- **PyVis**: gestisce rappresentazioni interattive di reti anche molto complesse

3 Creazione della rete

Per creare la rete siamo partiti dai seguenti 5 utenti, scaricandone follower e following:

- **@mizzaro** - 156 Follower - 331 Following
- **@damiano10** - 785 Follower - 836 Following
- **@Miccighel_** - 331 Follower - 211 Following
- **@eglu81** - 540 Follower - 621 Following
- **@KevinRoitero** - 103 Follower - 256 Following

Per ciascuno degli utenti sopra poi, sono stati scelti 5 followers e 5 followings a caso, e per ognuno di questi sono stati scaricati rispettivamente 10 loro followers e 10 loro followings.

3.1 Inizializzazione API di Twitter

Questa fase di inizializzazione prevede di creare un utente sulla piattaforma Twitter e di richiedere l'abilitazione all'utilizzo delle API, al fine di ricevere i parameteri necessari per iniziare ad utilizzarle. Maggiori informazioni sulla procedura sono disponibili sulla documentazione `https://developer.twitter.com/en/docs/getting-started`

3.2 Download e serializzazione dei dati

Tramite Tweepy abbiamo scaricato i dati necessari per la realizzazione della rete. Abbiamo deciso di salvare i dati scaricati in formato JSON all'interno della subdirecotry */data* del progetto. Al fine di agevolare le operazioni di lavoro sui file JSON abbiamo realizzato alcune funzioni dedicate, descritte nel Jupiter Notebook.

3.3 Creazione del grafo

Di seguito alcuni frammenti di codice esemplificativi della procedura di creazione

Inizializzazione grafo

```
1     twtNet = nx.DiGraph(group_members= ["Gasparollo Denis", "Francescut Matteo", "
          Bensi Martino", "Rizzetto Sasha"])
```

Aggiunta dei nodi

```
1  for node in nodes:
2          twtNet.add_node(node["id"],
3                          name= node["name"],
4                          screen_name= node["screen_name"],
5                          location= node["location"],
6                          description= node["description"],
7                          followers_count= node["followers_count"])
```

Segue il frammento di codice che aggiunge un arco presa in analisi una coppia di nodi

Aggiunta degli archi

```
1  if "followings" in splitedFileName[len(splitedFileName) - 1]:
2      twtNet.add_edge(idDictionary[splitedFileName[0]], user["id"])
3  elif "followers" in splitedFileName[len(splitedFileName) - 1]:
4      twtNet.add_edge(user["id"], idDictionary[splitedFileName[0]])
```

NOTA: **idDictionary** è un dizionario composto da una coppia *screen_name* - *id*

3.4 Creazione della mappa interattiva

Realizziamo ora con l'ausilio di PyVis la mappa interattiva della rete.

Creazione mappa PyVis

```
net.barnes_hut()
net.from_nx(twtNet)
neighbor_map = net.get_adj_list()
for node in net.nodes:
    node["value"] = len(neighbor_map[node["id"]])
```

3.5 Creazione del sottografo

Con il seguente frammento di codice creiamo il sottografo scegliendo come nodo centrale l'utente KevinRoitero. Successivamente calcoliamo la cricca massima.

Creazione sottografo

```
roiteroGraph = nx.ego_graph(twtNet, 3036907250, radius=1)
clique = clq.max_clique(nx.to_undirected(roiteroGraph))
```

La dimensione della cricca massima risultante è 5.
Eseguire la sezione specifica del notebook per visualizzare la rappresentazione grafica della cricca calcolata.

4 Analisi della rete

Sulla rete ottenuta abbiamo effettuato le seguenti analisi

4.1 Verifiche di *connessione* e di *bipartizione*

Il grafo risulta ovviamente **connesso**, seppur debolmente. Inoltre, come ci si aspettava, **non è bipartito**.

4.2 Misurazioni di *distanze*: centro, diametro e raggio

Il **centro** del grafo è rappresentato dai seguenti 3 nodi: *damiano10*[132646210], *eglu81*[19659370], *KevinRoitero*[3036907250]. Il risultato non sorprende dato che la rete è stata creata a partire dai 5 utenti forniti inizalmente. L'unico esito non propriamente previsto è *KevinRoitero*, ci aspettavamo il terzo fosse *Miccighel_*.
Il **diametro** risultante è **6**, il che ci fonisce una prima conferma del fenomeno "small-world". Infatti il valore è esattamente lo stesso della teoria dei "6 gradi di separazione".
Infine il **raggio** calcolato è **3**

4.3 Calcolo della *copertura minima* degli archi

```
minCover = nx.min_edge_cover(nx.to_undirected(twtNet))
```

Per motivi di spazio non riportiamo qui il risultato, ma rimandiamo al Jupiter Notebook.

4.4 Misurazioni di *centralità*: Betweenness centrality, Closeness centrality, Degree centrality, In-degree centrality, Out-degree centrality, Page Rank e HITS

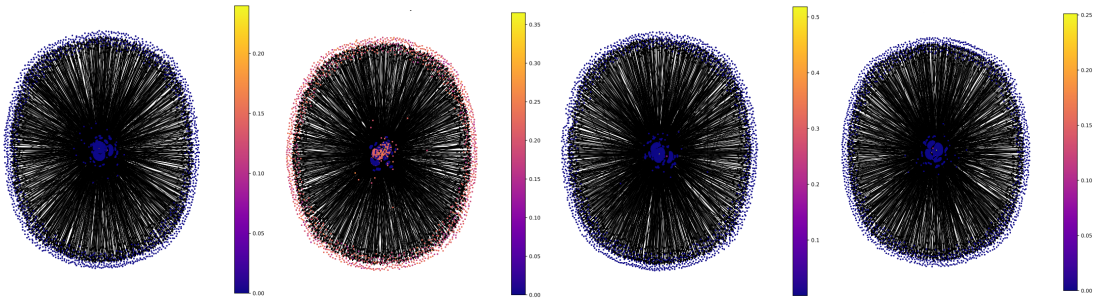
I grafici e le tabelle qui presenti sono meglio visibili sulla repository all'interno della directory */centrality*

	between_centrality	closeness_centrality	degree_centrality	in_degree_centrality	out_degree_centrality	page_rank	hits_hub	hirs_authorities
18932422	0.053671	0.275478	0.155541	0.050144	0.105398	0.023708	0.026042	0.002456
132646210	0.239781	0.365178	0.518045	0.251038	0.267007	0.121508	0.069893	0.006888
15750573	0.112468	0.274166	0.173746	0.106036	0.067710	0.058212	0.002266	0.001397
19659370	0.173588	0.338558	0.370489	0.172149	0.198339	0.084725	0.037860	0.004365
3036907250	0.048493	0.265937	0.114660	0.032897	0.081763	0.015357	0.017292	0.001577
...	...	...	...	...	...	...	...	...
797863519953764353	0.000000	0.000000	0.000319	0.000000	0.000319	0.000130	0.000022	0.000000
1251182739035795463	0.000000	0.000000	0.000319	0.000000	0.000319	0.000130	0.000022	0.000000
1281172664896360449	0.000000	0.000000	0.000319	0.000000	0.000319	0.000130	0.000022	0.000000
1284176128534409216	0.000000	0.000000	0.000319	0.000000	0.000319	0.000130	0.000022	0.000000
1319272146963673089	0.000000	0.000000	0.000319	0.000000	0.000319	0.000130	0.000659	0.000000

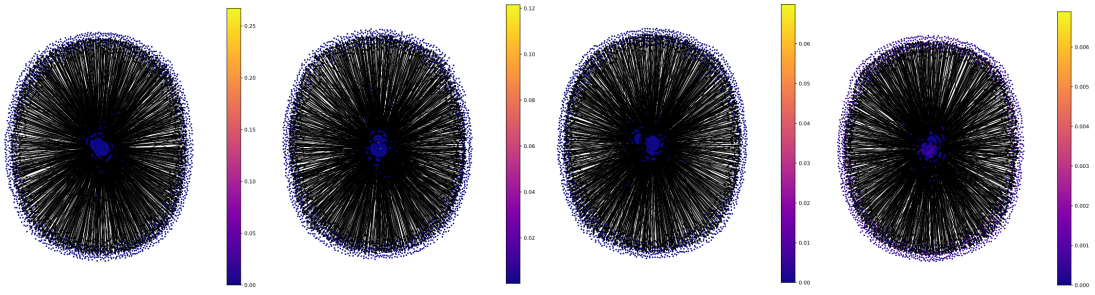
Rappresentazioni delle centralità sul grafo

Funzione draw per i grafici (es. Betweenness centrality)

```
1 draw(twtNet,
2       nx.spring_layout(twtNet),
3       betweennessCentrality,
4       "Betweenness centrality")
```



(a) Betweenness centrality (b) Closeness centrality (c) Degree centrality (d) In-degree centrality



(e) Out-degree centrality (f) Page Rank (g) HITS (hubs) (h) HITS (authorities)

4.5 Osservazione del fenomeno *"small-worldness"* mediante il calcolo di Coefficiente Sigma e Coefficiente Omega

Risultato del **coefficiente Omega**: *0,042*
Risultato del **coefficiente Sigma**: *0,841*
I risultati ottenuti ci inducono ad affermare, come ci aspettavamo visto il risultato del diametro, che la rete possiede le caretteristiche di "small-worldness". Più nel dettaglio il coefficiente omega supporta pienamente questa ipotesi, essendo molto prossimo allo zero. Il coefficiente sigma invece, seppur prossimo all'uno, risulta abbastanza contraddittorio. Infatti il valore atteso per una rete "small-world" dovrebbe essere superiore a 1. Ipotizziamo che lo scostamento verificato possa essere imputabile alle semplificazioni apportate alla rete per motivi di limitazioni computazionali.

4.6 Calcolo della *correlazione fra le misure di centralità* tramite coefficiente di Pearson e coefficiente di Kendall

Figura 2: Correlazioni di Pearson

	between_centrality	closeness_centrality	degree_centrality	in_degree_centrality	out_degree_centrality	page_rank	hits_hub	hirs_authorities
between_centrality	1.000000	0.065170	0.993015	0.996508	0.975441	0.997209	0.908778	0.346496
closeness_centrality	0.065170	1.000000	0.075923	0.096306	0.056485	0.085625	0.029402	0.727422
degree_centrality	0.993015	0.075923	1.000000	0.991489	0.993120	0.987748	0.944820	0.368928
in_degree_centrality	0.996508	0.096306	0.991489	1.000000	0.969422	0.998782	0.912141	0.379209
out_degree_centrality	0.975441	0.056485	0.993120	0.969422	1.000000	0.963467	0.960480	0.354319
page_rank	0.997209	0.085625	0.987748	0.998782	0.963467	1.000000	0.899004	0.363526
hits_hub	0.908778	0.029402	0.944820	0.912141	0.960480	0.899004	1.000000	0.376567
hirs_authorities	0.346496	0.727422	0.368928	0.379209	0.354319	0.363526	0.376567	1.000000

Figura 3: Correlazioni di Kendall

	between_centrality	closeness_centrality	degree_centrality	in_degree_centrality	out_degree_centrality	page_rank	hits_hub	hirs_authorities
between_centrality	1.000000	0.139118	0.352842	0.254479	0.265926	0.214026	0.161398	0.144691
closeness_centrality	0.139118	1.000000	0.460035	0.791250	-0.314397	0.820417	-0.202221	0.961653
degree_centrality	0.352842	0.460035	1.000000	0.587766	0.408041	0.538018	0.376041	0.514347
in_degree_centrality	0.254479	0.791250	0.587766	1.000000	-0.371451	0.828113	-0.270011	0.810806
out_degree_centrality	0.265926	-0.314397	0.408041	-0.371451	1.000000	-0.267117	0.812326	-0.280521
page_rank	0.214026	0.820417	0.538018	0.828113	-0.267117	1.000000	-0.185083	0.821294
hits_hub	0.161398	-0.202221	0.376041	-0.270011	0.812326	-0.185083	1.000000	-0.170879
hirs_authorities	0.144691	0.961653	0.514347	0.810806	-0.280521	0.821294	-0.170879	1.000000

5 Conclusione

L'esperimento è stato limitato principalmente da questioni correlate a restrizioni computazionali, di memory usage e delle API di Twitter.
I risultati riportati non possono pertanto ritenersi significativi, bensì indicativi.
Tuttavia l'analisi di questa rete sociale ci ha portato a riflettere su diversi aspetti, primo tra i quali il concetto di "small-worldness". Effettivamente abbiamo avuto una riconferma abbastanza forte in questo senso tramite le misurazioni delle distanze e dei coefficienti Omega e Sigma, cosa che ci aspettavamo per una rete sociale.