Primo Progetto di Social Computing

Massimiliano Baldo 142296
Simone Dalla Pietà 141995
Ilaria Fenos 142494
Emanuele Lena 142411

A.A. 2020/21

Indice

1 Introduzione

1.1 Obiettivi del progetto

L'obiettivo del progetto è di reperire una porzione della rete sociale del social network Twitter, per poi fare un'analisi applicando alcune delle più tipiche tecniche di studio dei grafi.
Nel dettaglio, si intende:

- Reperire i dati pubblici di 5 di profili di partenza, dei profili a loro direttamente correlati (followers e followed) e di ulteriori profili casuali (scelti secondo certi criteri spiegati in seguito)
- Costruire un grafo che rappresenta la rete sociale, dove
  - i nodi sono i profili scaricati
  - gli archi (diretti) indicano una relazione di follower → following ("chi segue chi")
- Applicare le più comuni tecniche di analisi sul grafo, quali la visualizzazione del grafo, misura delle distanze e centralità, calcolo della copertura minima e la stima della "small-world-ness" del grafo.
- Calcolo delle correlazioni tra le variabili calcolate

1.2 Strumenti e tecnologie

Si usano i seguenti strumenti e tecnologie:

- API di Twitter - per il reperimento dei dati necessari
- Linguaggio Python - per la semplicità d'uso
- Librerie (principali) del linguaggio Python: Tweepy, NetworkX, Pandas, Pyvis
- Strutture di supporto Cloud: Google Colaboratory per il codice e Google Drive per il salvataggio dei dati

2 Reperimento dei dati

2.1 Metodologie di reperimento, rappresentazione e divisione del carico di lavoro

2.1.1 Metodologie di reperimento

Il reperimento dei dati è stato effettuato tramite l'interrogazione degli endpoint di Twitter con il supporto della libreria Tweepy. Alcune funzioni vengono eseguite con il supporto di Cursor (strumento di Tweepy che gestisce automaticamente la paginazione dei risultati).

2.1.2 Metodologie di rappresentazione

Si è scelto di rappresentare i dati reperiti mediante dei dataframe Pandas. Nel dettaglio, le informazioni sono state rappresentate tramite questi due formati:

- Il primo formato - d'ora in poi chiamato "df_users" - rappresenta i dati dei singoli profili, un profilo per riga; ogni colonna corrisponde ad un campo dell'oggetto ritornato da api.get_user, per esempio:

  - la colonna “id” contiene il codice identificativo univoco del profilo,
  - la colonna “screen_name” contiene l’username (univoco) del profilo,
  - la colonna “followers_count” contiene il numero di seguaci (“followers”) del profilo;

- Il secondo formato - d’ora in poi chiamato “df_relations” - rappresenta tutte le relazioni di tipo “Follower segue Following” rilevate durante l’esplorazione.
  - ogni riga indica una relazione del tipo “A segue B”;
  - le colonne “Follower” e “Following” rappresentano rispettivamente i profili “che seguono” e “quelli seguiti”; ogni profilo si identifica con il suo id.

A questi due formati principali si aggiunge un terzo formato - d’ora in poi chiamato “df_accurate_relations” - utile a rappresentare i risultati delle interrogazioni effettuate con api.show_friendship.

- Ogni riga riporta i dettagli della relazione tra 2 profili (“source” e “target”)
- Oltre alle colonne “source” e “target”, ci sono altre 2 colonne “follow” e “followed _by”, che indicano rispettivamente se “source” segue “target” e se “source” è seguito da “target”.

Tutti questi dataframe sono stati salvati come dataset csv su una cartella Drive.

2.1.3 Divisione del carico di lavoro

Viste le limitazioni di interrogazioni effettuabili tramite le API di twitter, si è diviso il carico di lavoro delle interrogazioni su 4 notebook Colaboratory differenti, che - con lo stesso codice - interrogano gli endpoint con chiavi d’accesso differenti in parallelo.

Ogni notebook ha quindi generato più datasets df_users e df_relations. Questi datasets sono stati poi uniti in un 2 unici datasets finali attraverso un opportuno codice che concatena i dataset e rimuove eventuali duplicati.

Si sono salvati i due dataset finali df_users e df_relations, sempre nel formato csv.

2.2 Reperimento dei profili principali e dei profili direttamente connessi

Si è partiti dal reperimento dei dati di 5 profili principali. Di questi profili si sono reperiti tutti i dati pubblici con la chiamata di api.get_user. Dopo di che, si sono reperiti anche i dati dei loro follower e following tramite le chiamate api.followers e api.friends. Durante il reperimento dei dati di followers e followed, si riempie anche il dataframe df_relations con le informazioni su che relazione hanno quest’ultimi con il profilo principale:

- Se un profilo A è follower del profilo principale X, si inserirà una riga “A segue X”;
- Se un profilo B è followed (friend) di X, si inserirà una riga “X segue B”.

5 profili principali: @mizzaro, @damiano10, @Miccighel_, @eglu81, @KevinRoitero

2.3 Reperimento dei profili casuali aggiuntivi

Per ognuno dei 5 profili principali, si scelgono 5 followers e 5 followed casuali. Come criterio di scelta, si considerano solo i followers con almeno 10 followers e i followed con almeno 10 followed (i profili si trovano cercando nel dataframe df_relations).

Per ognuno dei 5 followers casuali si reperiscono gli ids dei suoi followers con api.followers_ids.

Di questi ids, si scelgono casualmente 10, si recuperano i dati dei profili con tali ids (tramite api.get_user) e si annota su df_relations di chi sono followers.

Si effettua una procedura analoga per i 5 followed casuali scelti e 10 loro followed casuali.

2.4 Verifica della relazione tra i profili

Si vuole verificare la relazione tra tutti i profili scaricati e i 5 iniziali. Per verificare la relazione, si chiama la funzione api.show_friendship tra ognuno dei profili ed ognuno dei 5 iniziali (prendendo le opportune precauzioni per non verificare 2 volte la stessa relazione o verificare la relazione del profilo con se stesso). Viste le limitazioni di richieste sugli endpoint, confrontare tutti i profili sarebbe risultato molto dispendioso.

Si è giunti però alla conclusione, che già si conoscono le relazioni con i 5 profili iniziali:

- Se un profilo è follower di uno dei 5, si individua durante la chiamata api.followers;
- Se un profilo è invece seguito da uno dei 5, si individua durante la chiamata api.friends

Tutte queste informazioni, si ipotizza siano in df_relations. Per dimostrare ciò, si prova a chiamare api_show_friendship solo su un campione di 100 profili, che vengono confrontati con i 5 principali. Effettuata questa interrogazione (e prodotto un dataset del tipo df_accurate_relations), si effettua una verifica:

- se in df_accurate_relations, è segnato che un profilo segue un altro, si controlla che la relazione sia presente in df_relations;
- se in df_accurate_relations, invece è segnato che un profilo non segue un altro, si controlla che la relazione non sia presente in df_relations.

A verifica terminata, si constata che in df_relations ci sono già tutte le informazioni, quindi non è necessario proseguire con la chiamata di api_show_friendship per tutti i profili.

3 Analisi della rete sociale

3.1 Costruzione del grafo della rete sociale

Si riproduce la rete sociale attraverso la libreria Python Networkx. Si crea un grafo diretto con le seguenti caratteristiche:

- nei nodi i dati di tutti gli profili scaricati, cioè tutti i profili nel dataset df_users;
- gli archi (diretti) indicano le relazioni tra i profili (se A segue B allora si inserisce un arco diretto da A a B); per conoscere le relazioni, si fa riferimento al dataset df_relations.

Si ricava anche una versione del grafo dove gli archi non sono diretti, necessaria in seguito per alcune analisi.

3.2 Panoramica generale della rete

3.2.1 Caratteristiche generali

Il grafo generato è composto da 3102 nodi e 4648 archi e presenta le seguenti caratteristiche:

- è connesso (nx.is_connected);
- non è bipartito (nx.is_bypartite);
- ha come centro (nx.center) 3 nodi:@KevinRoitero, @eglu81, @damiano10;
- un diametro equivalente a 6 (nx.diameter) e un raggio equivalente a 3 (nx.radius).

3.2.2 Visualizzazione della rete

Si allega una rappresentazione della rete prodotta con la libreria pyvis.

3.3 Misure della centralità

3.3.1 Risultati

Si calcolano, per ogni profilo, le seguenti misure di centralità:

- Degree, In Degree, Out Degree (con i metodi nx.degree_centrality, nx.in_degree_centrality e nx.out_degree_centrality);
- Betweenness e Closeness Centrality (con i metodi nx.betweenness_centrality e nx.closeness_centrality);
- Pagerank e HITS - quindi hubness e centrality (con i metodi nx.pagerank e nx.hits).

Quello che risulta è che l'utente Damiano Spina (132646210) è l'utente che ha valore più alto in tutte le misure di centralità, questo non sorprende poiché Damiano è un punto centrale della rete con molti followers e ciò sicuramente influisce sul suo "status".

3.3.2 Correlazione tra le misure effettuate

Si calcolano le correlazioni di Pearson e Kendall tra le diverse misure di centralità effettuate, riassumendo i risultati in due tabelle (allegate in csv). Osservando le tabelle di correlazioni, si possono alcuni aspetti interessanti di questa rete, in particolare:

- Nella tabella rappresentata la correlazione di Pearson, si osserva una forte correlazione tra tutte le misure di centralità (ad eccezione per le authority). Questo si spiega dal fatto che la rete intera è costruita principalmente su 5 utenti, quindi:
  - degree_centrality, in_degree_centrality ed out_degree_centrality saranno indubbiamente alti rispetto agli altri nodi; di conseguenza, anche il punteggio con pagerank risulterà essere alto;
  - i nodi risultano essere quasi dei "centri stella" nella rete, avranno quindi una betweenness_centrality molto alta;
  - visto che tutti i profili raggiunti si trovano ad una distanza massima di 2 da almeno uno dei 5 profili principali, questi tenderanno ad avere anche una closeness_centrality alta.
- Nella tabella rappresentate la correlazione di Kendall, si osserva una correlazione tra gli hubness e gli out-degree (attorno a 0.8); la cosa era prevedibile, in quanto gli hubness si basano sugli out-degree dei nodi; alla stessa maniera, se si guarda gli authority con gli in-degree anche li la correlazione è alta (sempre attorno a 0.8).
- Infine, sempre sulla tabella di Kendall, si osserva una correlazione alta tra pagerank e gli in-degree (sempre attorno a 0.8); anche questo era prevedibile, in quanto pagerank e autority dipendono entrambi dall'in_degree.

3.4 Calcolo della cricca massima

3.4.1 Calcolo del sotto-grafo ridotto

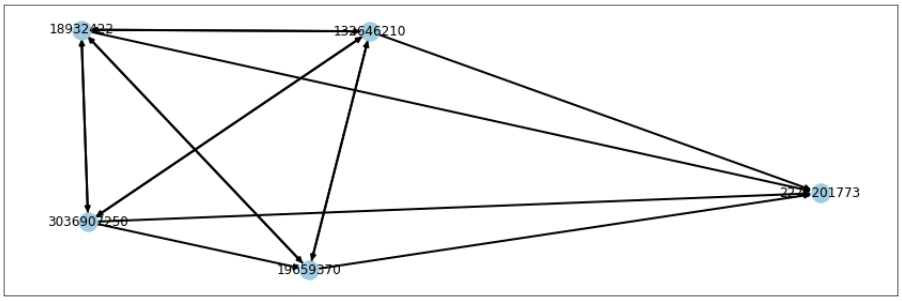
Si vuole calcolare la cricca massima della rete sociale. In quanto l'operazione risulterebbe particolarmente dispendiosa, si decide di calcolare la cricca massima soltanto di una porzione della rete.
Si sceglie come porzione campione l'ego-grafo (grafo composto da tutti i nodi direttamente collegati ad esso) del nodo del profilo @KevinRoitero, che si ricava con la funzione nx.ego_graph (sul grafo non diretto). Si ottiene un grafo composto da 310 nodi e 639 archi.

3.4.2 Calcolo della cricca massima

Con nx.algorithms.approximation.clique.large_clique_size, si ricava la dimensione della cricca massima (6). Con nx.algorithms.approximation.clique.max_clique, si ricava la cricca massima, che è composta dai seguenti nodi:

@damiano10, @mizzaro, @eglu81, @SIGIRForum, @KevinRoitero.

Visualizzazione della cricca massima:



3.5 Calcolo della copertura minima

Si ricava l'albero di copertura minima (sempre sull'ego-grafo già ricavato) attraverso la funzione nx.min_edge_cover.

3.6 Stima della "small-world-ness" del grafo

Lo "small-world-ness effect" ("effetto piccolo mondo") è un fenomeno che si può osservare all'interno di una rete nella quale i nodi sono strettamente connessi tra di loro e le distanze dei percorsi che li interconnettono sono relativamente brevi.

Per misurare l'effetto, si ricavano e si analizzano i coefficienti Omega e Sigma (per ricavare, si eseguono i metodi nx.omega ed nx.sigma sul grafo indiretto). Si veda la documentazione di Networkx per vedere come vengono calcolati:

- Il risultato previsto per Omega è un valore compreso tra 1 e -1; se vicino allo zero significa che si ha small-world-ness. Nel nostro caso, si ricava il valore 0.0029;
- Per Sigma invece, si dice che si ha small-world-ness se è > 1. Nel nostro caso, si ricava il valore 0.94, che non è propriamente > 1, ma si avvicina molto.

Considerando i risultati osservati per Sigma e Omega, si può affermare che nella rete analizzata si verifica l'effetto piccolo mondo.

4 Bibliografia e sitografia

- lezioni e slide del prof. Soprano Micheal (sito ufficiale: `https://michaelsoprano.com/`)
- lezioni e slide del prof. Mizzaro Stefano (sito ufficiale: `http://users.dimi.uniud.it/~stefano.mizzaro/`)
- documentazione ufficiale Tweepy: `http://docs.tweepy.org/en/latest/`
- documentazione ufficiale Networkx: `https://networkx.org/documentation/stable/index.html`
- documentazione ufficiale Pandas: `https://pandas.pydata.org/docs/`
- documentazione ufficiale PyVis: `https://pyvis.readthedocs.io/en/latest/`