

Primo Progetto di Social Computing Reperimento e Analisi di una Rete Sociale

Massimiliano Baldo 142296

Simone Dalla Pietà 141995

Ilaria Fenos 142494

Emanuele Lena 142411

A.A. 2020/21

Indice

1	Introduzione	1
1.1	Obiettivi del progetto	1
1.2	Strumenti e tecnologie	1
2	Reperimento dei dati	1
2.1	Metodologie di reperimento, rappresentazione e divisione del carico di lavoro . .	1
2.1.1	Metodologie di reperimento	1
2.1.2	Metodologie di rappresentazione	1
2.1.3	Divisione del carico di lavoro	2
2.2	Reperimento dei profili principali e dei profili direttamente connessi	2
2.3	Reperimento dei profili casuali aggiuntivi	2
2.4	Verifica della relazione tra i profili	3
3	Analisi della rete sociale	3
3.1	Costruzione del grafo della rete sociale	3
3.2	Panoramica generale della rete	3
3.2.1	Caratteristiche generali	3
3.2.2	Visualizzazione della rete	4
3.3	Misure della centralità	4
3.3.1	Risultati	4
3.3.2	Correlazione tra le misure effettuate	4
3.4	Calcolo della cricca massima	5
3.4.1	Calcolo del sotto-grafo ridotto	5
3.4.2	Calcolo della cricca massima	5
3.5	Calcolo della copertura minima	5
3.6	Stima della “small-world-ness” del grafo	5
4	Bibliografia e sitografia	6

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 \rightarrow 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.

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 “following” e “followed_by”, che indicano rispettivamente se “source” segue “target” e se “source” è seguito da “target”.

Tutti questi dataframe, una volta costruiti, si salvano in formato csv su una cartella Google 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 dataset `df_users` e `df_relations`. Questi dataset sono stati poi uniti in un 2 unici dataset finali attraverso un opportuno codice che unisce i dati ed elimina i 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 l’esecuzione di `api.get_user`. Dopo di che, si sono reperiti anche i dati dei loro followers e following tramite le chiamate `api.followers` e `api.friends`. Durante il reperimento dei dati, viene riempito anche il dataframe `df_relations` con le informazioni su che relazioni ci sono tra questi profili e il profilo principale da cui vengono scoperti:

- 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 (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 dovrebbe chiamare 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). Però viste le limitazioni di richieste sugli endpoint, confrontare tutti i profili sarebbe risultato molto dispendioso.

Fortunatamente, si è giunti 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`.

Quindi si ipotizza che tutte queste informazioni si trovino già in `df_relations`. Per dimostrare ciò, si prova a chiamare `api_show_friendship` 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 si inseriscono i dati di tutti gli profili scaricati, cioè tutti i profili presenti 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 (verificato con `nx.is_connected`);

- non è bipartito (verificato con `nx.is_bipartite`);
- ha come centro (individuato con `nx.center`) 3 nodi: `@KevinRoitero`, `@eglu81`, `@damiano10`;
- un diametro equivalente a 6 (calcolato con `nx.diameter`) e un raggio equivalente a 3 (calcolato con `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 authority (con i metodi `nx.pagerank` e `nx.hits`).

Si osserva che l'utente `@damiano10` (con id `132646210`) ha valore più alto in tutte le misure di centralità, questo non sorprende poiché `@damiano10` è un nodo 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 notano alcuni aspetti interessanti; 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 attorno a 5 utenti di cui si reperiscono tutti i nodi adiacenti, 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 hubness e out-degree (attorno a 0.8); la cosa era prevedibile, in quanto hubness si basa sull'out-degree dei nodi; alla stessa maniera, anch'è la correlazione tra authority e in-degree è alta (sempre attorno a 0.8).
- Infine, sempre sulla tabella di Kendall, si osserva una correlazione alta tra pagerank e in-degree (sempre attorno a 0.8); anche questo era prevedibile, in quanto pagerank e authority 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.

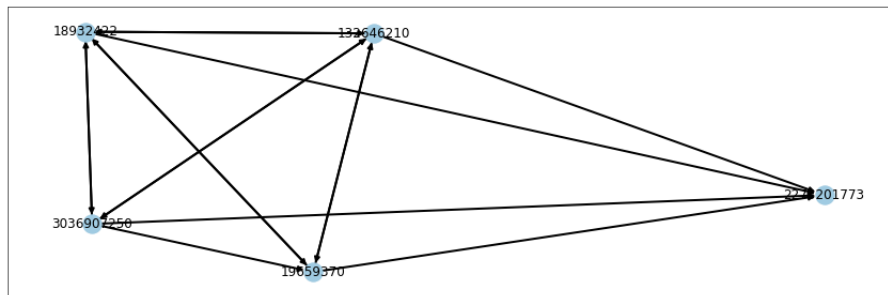
Si sceglie come porzione campione l'ego-grafo (grafo composto da tutti i nodi direttamente collegati ad esso) del profilo @KevinRoitero, che si ricava con la funzione `nx.ego_graph` (sul grafo non diretto). Si ottiene un grafo composto da 310 nodi e 639 archi.

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 e le distanze tra di essi sono brevi.

Per misurare l'effetto, si ricavano e si analizzano i coefficienti Omega e Sigma (che si calcolano eseguendo i metodi `nx.omega` e `nx.sigma` sul grafo indiretto, si veda la documentazione di Networkx per maggiori informazioni):

- 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/>