Relazione Progetto 1, Social Computing

Gianluca Fava - 158157 - 158157@spes.uniud.it
Alessando Ligugnana - 158038 - 158038@spes.uniud.it
Enrico Paravano - 158140 - 158140@spes.uniud.it
Davide Domenico Tammaro - 157795 - 157795@spes.uniud.it

Dicembre 2023

1 Executive summary

In questo progetto si richiedeva di utilizzare la libreria Python SerpAPI e Pandas per estrarre informazioni da Google Scholar su 7 autori. In particolare, si ottenevano dati come ID, numero di citazioni, interessi e coautori, aggregando poi queste informazioni in un dataframe `nodes.csv` e salvando la relazione di co-authorship tra i 7 autori originali e i loro coautori in un dataframe `edges.csv`. Successivamente, si richiedeva di creare un grafo indiretto rappresentante le relazioni di co-authorship tra gli autori, di colorare i nodi in base al grado e di produrre visualizzazioni interattive e poi calcolarne diverse misure di rete.

L'espansione del grafo ha portato ad un aumento della centralità dei nodi, formazione di cluster più evidenti ed una diminuzione della distanza media, riflettendo l'influenza del preferential attachment e cambiamenti nelle metriche di centralità degli autori.

2 Metodologia

2.1 Scaricamento Dei Dati

Per scaricare i dati sono state usate la libreria Python SerpAPI e 3 funzioni:

1. `author_profile(apiKey, name, affiliation)`: questa funzione viene usata per trovare le informazioni da inserire in `nodes.csv` sia per i 7 autori originali che per i loro coautori. Nello specifico la funzione prende in input l'`apiKey` e il `nome` (e cognome) di un autore, la sua `affiliazione`. Fa una chiamata SerpAPI a GoogleScholar usando come parametro di ricerca il nome e salva la risposta in json; è importante notare che ci potrebbero essere più autori con lo stesso nome. Per ovviare a ciò si scorrono tutti i profili trovati e si controlla per ognuno se l'affiliazione passata in input è contenuta nell'affiliazione di quel profilo se così fosse si salva l'indice del profilo e si interrompe lo scorrimento, altrimenti vuol dire che il profilo non ha affiliazione e quindi si prende semplicemente il primo autore con il nome uguale a quello dato in input. Avendo il profilo desiderato, si possono salvare `author_id` e `cited_by`, ricavate dalle informazioni nel json del profilo, invece per quanto riguarda le informazioni degli interessi dell'autore, (key `interests`) c'è da fare un controllo ulteriore perché non per tutti i profili è presente; nel caso in cui non sia presente viene creata e settata come stringa vuota. Infine viene passato a `update_nodes()` un dataframe avente una riga contenente tutti i dati (`name, affiliation, author_id, cited_by, interests`); per quanto riguarda l'affiliazione se quando sono stati scorsi i profili era stata trovata vuota, viene passata come stringa vuota.

2. `get_profile_coauthors(apiKey, authorID)`: questa funzione viene usata per trovare tutti i coautori dell'autore dato in input (gli verranno passati solo i 7 originali) e per ognuno di essi, richiama la funzione `author_profile` per salvare le informazioni del coautore; inoltre si salvano i collegamenti tra autori e coautori in `edges.csv`. Più nello specifico la funzione prende in input l'`apiKey` e l'`author_id` e fa una chiamata SerpAPI a GoogleScholar usando coma parametro di

ricerca `author_id`. La risposta conterrà, sotto la key `co_authors`, i valori di nome e l'affiliazione di ogni coautore (oltre che ad altri dati), quindi per ogni coautore si verifica se la coppia autore-coautore oppure coautore-autore è già presente in `edges.csv` (siccome si deve creare un grafo indiretto basta che la coppia sia presente in una sola direzione). Per salvare questa informazione si da alla variabile `notFoud` il valore `False` se è già presente altrimenti `True` e in questo caso viene concatenato ad `edges.csv` l'arco che rappresenta la coppia. Sempre per ogni coautore si deve richiamare la funzione `author_profile()` inserendo come affiliazione la propria, se esiste la key `affiliation` e non è vuota, oppure una stringa vuota se è presente la key ma non contiene nulla altrimenti se proprio non è presente la key, viene creata e inizializzata come stringa vuota. Infine, una volta che tutti i coautori sono stati inseriti in `nodes.csv` con tutti i campi popolati, quelli possibili (`name, affiliation, author_id, cited_by, interests`), tramite la funzione `author_profile()` viene sovrascritto `edges.csv` con quello appena creato contenente tutti i nuovi collegamenti autore-coautore.

3. `update_nodes(df_update)`: questa funzione viene usata per aggiungere un autore al dataframe `nodes.csv`. Più nello specifico prende in input il dataframe `df_update` creato dalla funzione `author_profile()` (contenente un solo autore con tutte le sue informazioni richieste da nodes: `name, affiliation, author_id, cited_by, interests`). Quindi si legge il `nodes.csv` salvato in locale che è aggiornato con tutti gli autori trovati fino a quel momento e si controlla se all'interno c'è un autore con nome uguale a quello presente in `df_update`. Se l'autore è già presente in `nodes.csv`, si cerca l'indice in cui è locato e si va sostituire la tupla al momento presente con quella data in input, `df_update` (perchè, come nel caso degli autori originali, potrebbe non essere completa). Se invece, l'autore non è ancora presente in `nodes.csv`, lo si concatena. In entrambi i casi alla fine il `nodes.csv` che è salvato in locale, viene sovrascritto con quello nuovo creato all'interno della funzione `update_nodes`.

Processo corretto di esecuzione: per far funzionare correttamente il programma bisognerebbe:

1. eseguire la sezione `"Creation of dataframe"` che legge il `nodes.csv` dato, contenente due colonne `name` e `affiliation`, per i 7 autori originali, ci aggiunge le colonne `author_id, cited_by, interests` inizializzate a `None` e lo sovrascrive a `nodes.csv`. Inoltre si crea e salva un nuovo dataframe `edges.csv` con due colonne `author1, author2`.

2. eseguire la sezione `"Main parte 1 & 2"` che legge `nodes.csv` e richiama `author_profile()` per i 7 autori originali in modo da trovarne tutte le informazioni mancanti (`author_id, cited_by, interests`). Poi sempre all'interno di questa sezione, si va a rileggere `nodes.csv`, che avrà le nuove informazioni sugli autori originali, e per ognuno richiamerà `get_profile_coauthors()` che troverà tutti i coautori degli autori dati e creerà il dataframe `edges.csv` contenente univocamente tutti gli archi autore-coautore. Per ogni coautore inoltre, richiamerà la funzione `author_profile()` che troverà le informazioni richieste (`author_id, cited_by, interests`) e aggiungerà i nuovi autori a `nodes.csv`.

2.2 Costruzione e Visualizzazione dei Grafi

Per la costruzione e la visualizzazione dei grafi sono state usate le librerie Python mtplotlib, NetworkX, Pickle e le funzioni:

1. `create_undirected_graph()`: questa funzione serve a generare un grafo indiretto vuoto utilizzando NetworkX. Per farlo, crea un grafo vuoto, legge `nodes.csv` e `edges.csv`, e ci aggiunge i nodi e gli archi letti. Infine, chiama la funzione `color_nodes()` per ottenere un dizionario di colori per i nodi dove ognuno è calcolato in base al proprio grado. Ritorna il grafo creato e il dizionario ritornato da `color_nodes()`.

2. `color_nodes(G)`: questa funzione calcola il grado dei nodi e crea un dizionario che associa il nome di un autore ad un colore. I nodi con grado 1 sono grigi, quelli con grado da 2 a 9 sono blu, da 11 a 20 sono viola, e gli altri sono gialli. Ritorna il dizionario creato.

3. `visualize_graph(graphData)`: questa funzione viene usata per visualizzare il grafo autori e coautori. Prende in input `graphData` che contiene un grafo non orientato rappresentato come un

grafo NetworkX (`graphData[0]`) ed un dizionario che associa i nomi degli autori ad un colore (`graphData[1]`).

4. `create_interactive_graph(graphID, graphData)`: questa funzione salva un grafico interattivo (HTML) degli autori e dei loro coautori e una copia del grafo viene salvata in formato gpickle. In particolare prende in input `graphID`, che identifica il tipo di grafo da creare (1=`first_graph`; 2=`extended_graph`) e `graphData` che contiene un grafo non orientato rappresentato come un grafo NetworkX (`graphData[0]`) ed un dizionario che associa i nomi degli autori ad un colore (`graphData[1]`). Utilizza l'algoritmo di Barnes-Hut per la disposizione dei nodi nel grafo, crea un grafico NetworkX dai dati passati con `graphData[0]`, imposta la dimensione dei nodi in base al loro grado e il loro colore seguendo `graphData[1]`. Infine salva la visualizzazione del grafo in HTML ed una sua immagine in PDF, salva i dati del grafo in un file di pickle. Il percorso e il nome dei file dipendono dall'identificativo del grafo (`graphID`): se è uguale a 1, i file sono salvati con il prefisso `"first_graph"`, altrimenti con `"extended_graph"`.

2.3 Espansione Del Grafo

Per l'espansione del grafo è stata utilizzato l'algoritmo del `preferential_attachment` della libreria NetworkX il quale si basa sul lavoro dei ricercatori Barabasi e Albert sulle reti Scale-Free.

Dal loro paper *Mean-field theory for scale-free random networks* del '99, per definire i modelli Scale-Free, il preferential attachment è sfruttato durante la crescita della rete di un nodo. In particolare, durante ogni passo di questa crescita, nella selezione dei nodi ai quali il nuovo nodo si connette, si basano sul principio dell'attaccamento preferenziale.

La probabilità $\prod$ che il nuovo nodo si colleghi al nodo i è modellata in funzione della connettività k_i di tale nodo, secondo l'espressione $\prod(k_i) = \frac{k_i}{\sum_j k_j}$.

Quindi sfruttando questo studio, per espandere il grafo, come già detto è stata usata la funzione della libreria Networkx `nx.preferential_attachment(G)` e sono state sviluppate due funzioni:

1. `preferential_attachment_top_n_edges(G, n)`: questa funzione calcola il preferential attachment di un grafo dato G creato dalla funzione `create_undirected_graph()` e restituisce i primi n archi aventi probabilità di creazione più elevata (in questo caso $n = 50$). Inizialmente, calcola il preferential attachment del grafo G utilizzando la funzione omonima di NetworkX che restituisce una lista di tuple, ognuna contenente due nodi e la probabilità di creazione di un arco tra di essi. Successivamente, vengono ordinati gli archi in base alla probabilità di creazione in ordine decrescente con la funzione `sorted` e inserendo il valore della probabilità come chiave di ordinamento. Dopodiché, vengono estratti i primi n archi con le probabilità più elevate, generando così una lista di coppie di nodi corrispondenti agli archi selezionati; infine si ritorna la lista creata.

   estende un grafo G aggiungendo 50 archi basati sull'attaccamento preferenziale. I nodi del grafo vengono colorati e viene restituito il grafo aggiornato insieme ai colori dei nodi. Il risultato viene salvato in formati HTML e PDF.

2. `extended_graph_preferential_attachment(G)`: questa funzione estende un grafo G aggiungendo 50 archi basati sul preferential attachment, i nodi del grafo vengono colorati e viene restituito il grafo aggiornato insieme ai colori dei nodi. In particolare, sfrutta la funzione `preferential_attachment_top_n_edges` per ottenere una lista di 50 archi con probabilità più elevate e li aggiunge al grafo G con il metodo `add_edges_from`. Successivamente, viene chiamata la funzione `color_nodes` per colorare i nodi del grafo. Infine, restituisce il grafo aggiornato insieme ai colori dei nodi. Queste operazioni permettono di estendere il grafo dato in input utilizzando l'algoritmo del preferential attachment e di ottenere una rappresentazione visuale con anche la colorazione dei nodi.

2.4 Ulteriori Assunzioni

Per la costruzione dei dataframe `nodes.csv` e `edges.csv` si è pensato di usare soltanto questi file salvati in locale senza l'utilizzo di un terzo dataframe di appoggio per contenente esclusivamente i coautori (come suggerito dalla traccia del progetto).

Si è fatto fatto ciò per garantire una maggiore pulizia e correttezza semantica del codice, dato che durante tutto il progetto si lavorava sempre sullo stesso file in locale e si aveva un unico "Main" che se eseguito poteva creare iterativamente tutto `nodes.csv` grazie alle funzioni realizzate.

Inoltre tutto il progetto è stato sviluppato mediate l'utilizzo di una repository su GitHub, per migliorare l'interoperabilità del gruppo, disponibile qui.

3 Risultati

3.1 Risultati parte 1

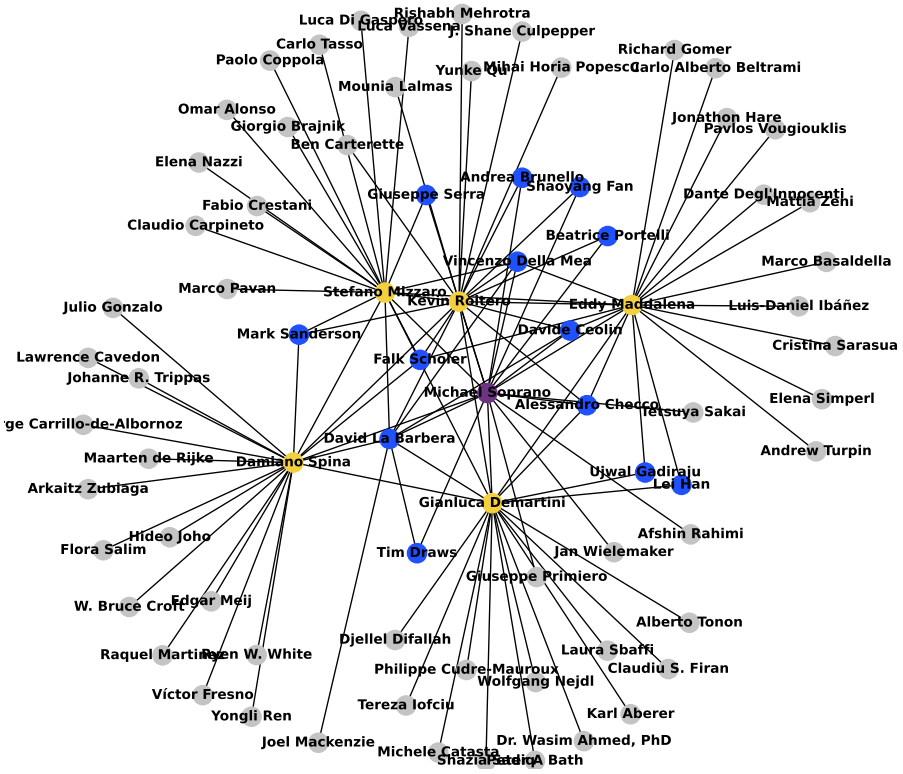
Durante l'implementazione della soluzione della prima consegna, sono stati ricavati i seguenti dati per 7 autori originali: `author_id, cited_by, interests`. Da questi dati si può notare che gli autori più citati sono Gianluca Demartini e Stefano Mizzaro.

3.2 Risultati parte 2

Nel seconda parte della consegna invece sono stati ricavati i seguenti dati per ciascun coautore degli autori originali: `name, affiliations, author_id, cited_by, interests`, e successivamente sono stati aggiunti nel file `nodes.csv`. Dopo aver fatto ciò, si avranno all'interno di `nodes.csv` ben 77 nodi tra autori e coautori, purtroppo alcuni di questi avranno il campo `affiliations` e/o `interests` vuoto.
Mentre analogamente all'interno di `edges.csv` sono stati salvati tutti gli archi diretti che rappresentano, le collaborazioni tra autori e coautori, le quali sono rappresentate dai 111 archi univoci, perché vengono salvati in una sola direzione.

3.3 Risultati parte 3

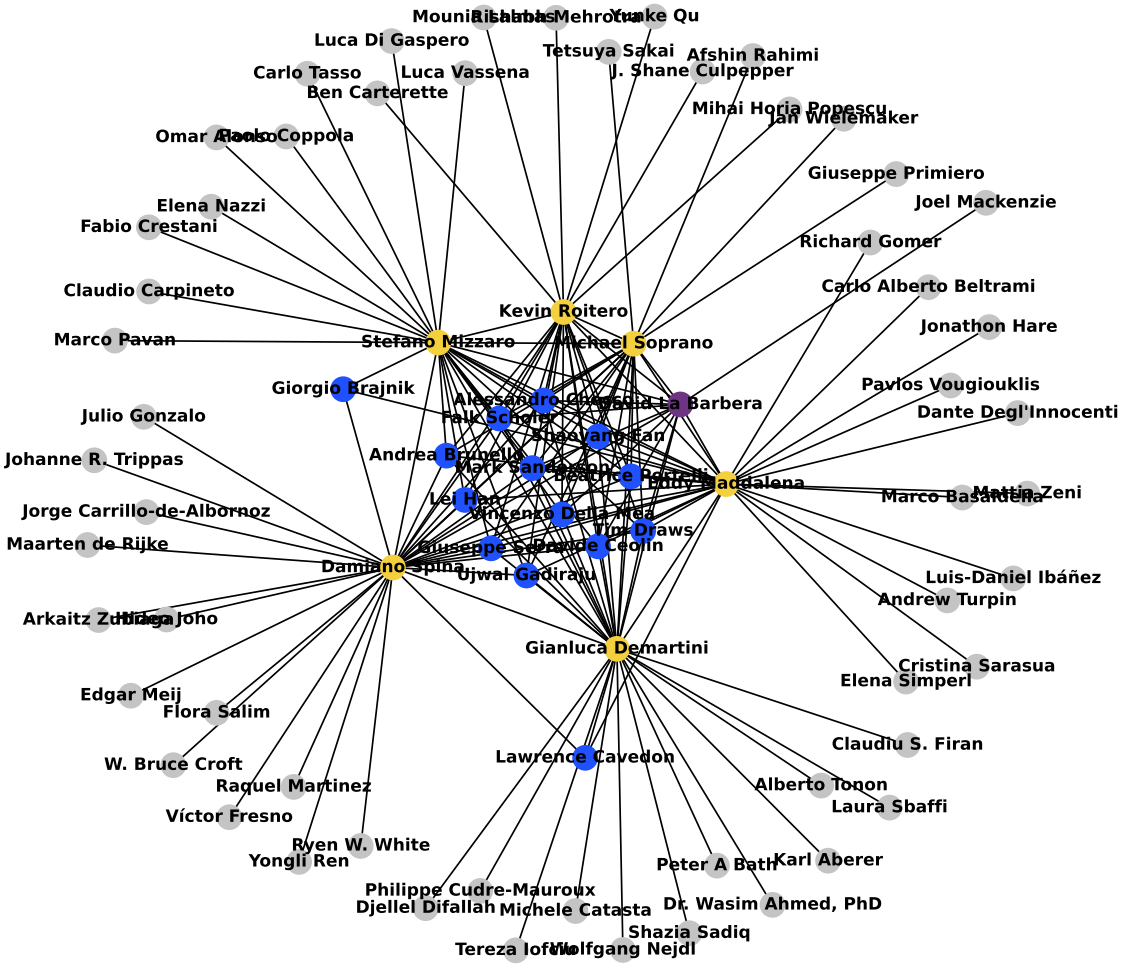
Nella terza parte è stato costruito il `first_graph` usando i nodi e gli archi ricavati nella sezione precedente.



La rappresentazione riportata qui sopra, è la raffigurazione grafica del `first_graph`, un grafo non orientato di autori-coautori, dove i nodi rappresentano gli autori e gli archi rappresentano le collaborazioni tra di essi. I nodi grigi sono di grado 1 mentre i nodi blu hanno un grado tra 2 e 9 invece i viola hanno grado tra 11 e 20 infine gli altri sono rappresentai in giallo. Si può notare che tra tutti i nodi, prevalentemente quelli con grado maggiore, sono quelli del gruppo dei 7 autori originali.

3.4 Risultati parte 4

Nella quarta parte è stato generato il grafo `extended_graph` partendo da quello precedentemente creato, aggiungendogli 50 archi generati seguendo il principio del preferential attachment.



Mentre questa è la rappresentazione grafica dell'`extended_graph` in cui, dopo l'applicazione del preferential attachment, gli autori che avevano già un grado elevato hanno ulteriormente aumentato il numero di archi ad essi collegati. Mentre per gli autori che avevano un grado basso è rimasto basso anche il numero di collegamenti a essi associati.

3.5 Risultati parte 5

Durante la risoluzione della quinta parte sono state calcolate le seguenti misure:

- **Coefficiente di clustering medio**: probabilità che i vicini di un nodo siano vicini tra loro. Il risultato nel primo grafo è circa 0.17, ciò significa che mediamente i nodi del grafo tendono a essere raggruppati in cluster, mentre nel grafo espanso il risultato è circa 0.21 e questo indica una tendenza dei nodi ad avere connessioni con i vicini, ma potrebbero esserci spazi meno densi nel grafo.

- **Centro del grafo**: indica i nodi del grafo che hanno eccentricità pari al raggio.
  Nel primo grafo vengono riportati i seguenti nodi: Michael Soprano, Kevin Roitero, Stefano Mizzaro, Gianluca Demartini; mentre nel grafo espanso vengono riportati i seguenti nodi: David La Barbera, Michael Soprano, Kevin Roitero, Stefano Mizzaro, Damiano Spina, Gianluca Demartini, Eddy Maddalena, Vincenzo Della Mea, Davide Ceolin, Tim Draws, Alessandro Checco, Falk Scholer, Mark Sanderson.

  Nel grafo espanso sono presenti molti più nodi centrali e ciò è dovuto all'algoritmo del preferential attachment che fa si che i nodi più connessi, a causa della loro popolarità passata (il grado del nodo), esercitano un'attrattivà maggiore per nuovi collegamenti.

- **Raggio**: misura la distanza minima tra un nodo centrale (o un insieme di nodi centrali) e tutti gli altri nodi della rete. In entrambi i grafi, il raggio è pari a 2, indicando la presenza di un gruppo limitato di nodi centrali, "hub", che consentono un accesso efficiente e facilitato a tutti gli altri nodi nella rete, facilitando la connettività complessiva.

  Dato che il raggio è influenzato principalmente dalla distanza tra i nodi più lontani e la struttura dei nodi centrali era già stabilita, l'aggiunta di soli archi con l'ausilio del preferential attachment non influenza il raggio ma rafforza soltanto le connessioni già presenti; quindi è probabile che questo valore rimanga invariato per i due grafi creati.

- **Distanza media**: questo valore rappresenta la lunghezza media del percorso più breve tra tutte le coppie di nodi nel grafo: $a = \sum_{s,t \in V} \frac{d(s,t)}{n(n-1)}$.

  Nel grafo iniziale, la distanza media è di circa 2.66, mentre nel grafo espanso si è ridotta a 2.40, questi valori sono relativamente bassi, e indicano una buona connettività e accessibilità nella rete. Questa diminuzione potrebbe essere dovuta al contributo positivo del preferential attachment nell'incrementare la connettività e nel rendere la rete più accessibile, riducendo la distanza media tra i nodi.

  È interessante notare che il raggio della rete rimane 2 in entrambi i grafi mentre la distanza media è diminuita, quindi, l'algoritmo del preferential attachment ha influenzato positivamente aggiungendo nuovi collegamenti e creando percorsi alternativi più brevi che hanno ridotto la distanza media. La discrepanza tra distanza media e raggio evidenzia la presenza di percorsi alternativi più lunghi tra alcune coppie di nodi le quali richiedono percorsi più lunghi rispetto alla connessione diretta con gli hub.

- **Transitività**: indica la presenza di relazioni indirette tra nodi, ovvero la presenza di triangoli nel grafo: $T = 3 \cdot \frac{\#triangoli}{\#triadi}$.

  Nel primo grafo la transitività è circa 0.15 e indica una bassa presenza di triangoli, invece nel grafo espanso il risultato è circa 0.29 indicando una maggiore propensione alla formazione di triangoli.

  Questo è dovuto dall'utilizzo del preferential attachment che durante l'aggiunta degli archi tende a collegare i nodi più connessi promuovendo la formazione di cluster più densi, portando quindi ad un aumento della transitività del grafo.

- **Coefficienti di small-world**:

  - **Omega**: indica se il grafo è a reticolo (se < 0) oppure è casuale (se > 0): $omega = \frac{Lr}{L} - \frac{C}{Cl}$.
    Nel primo grafo il valore di *omega* è circa -0.01 indicando che è un grafo a reticolo, mentre nel grafo espanso, il risultato è circa 0.003, quindi si può dedurre dal risultato che rappresenta un grafo casuale.
  - **Sigma**: indica se il grafo può essere rappresentato da un modello small-world (se > 1): $sigma = \frac{C}{Cr} - \frac{L}{Lr}$.
    Nel primo grafo, il coefficiente di small-world è circa 0.90, ciò indicando che non si può rappresentare il grafo sotto forma di modello small-world. Al contrario, nel grafo espanso, il coefficiente è di circa 0.99, che possiamo arrotondare a 1, deducendo che sia rappresentabile da un modello small-world.

Dove:
Lr = lunghezza media del percorso più breve in un grafo casuale equivalente;
L = lunghezza media del percorso più breve;
C = coefficiente di clustering medio;
Cl = coefficiente di clustering medio in un grafo a reticolo equivalente;
Cr = coefficiente di clustering medio di un grafo casuale equivalente;

3.6 Risultati parte 6

Nella sesta parte del progetto, per entrambi i grafi prodotti in precedenza, sono stati calcolati i seguenti indici per ciascun nodo:

- **Degree centrality**: numero di collegamenti che un nodo ha con altri nodi: $C_d^{norm}(v_i) = \frac{d_i}{n-1}$. Nel primo grafo gli autori Damiano Spina, Gianluca Demartini, Eddy Maddalena, Stefano Mizzaro, Kevin Roitero hanno il valore di degree centrality più alto rispetto al resto dei nodi ed è approssimabile per tutti ad un valore di circa 0.28, questi nodi sono quindi fortemente collegati ad altri nodi. Nel grafo espanso invece rimangono soltanto Damiano Spina, Eddy Maddalena e Gianluca Demartini come autori con la degree centrality più alta.
- **Betweenness centrality**: misura la frequenza di quanto un nodo in una rete si trovi lungo i cammini più corti tra coppie di altri nodi $c_B(v) = \sum_{s,t \in V} \frac{\sigma(s,t|v)}{\sigma(s,t)}$ dove: V è il set di nodi, $\sigma(s,t)$ è il numero di cammini più corti e $\sigma(s,t|v)$ è il numero di questi cammini, passanti per il nodi v. Nel primo grafo, Damiano Spina e Gianluca Demartini sono i nodi la cui betweenness centrality è più alta, mentre nel grafo espanso i nodi con i valori più alti sono Damiano Spina e Eddy Maddalena. Nel primo grafo ci sono 10 nodi con betweenness centrality maggiore di 0, mentre nel grafo espanso sono rimasti solamente in 7 nodi. Questa riduzione potrebbe essere attribuita all'utilizzo del preferential attachment, che ha introdotto nuovi archi e potenzialmente ridistribuito il flusso di informazioni, influenzando la centralità di alcuni nodi e modificando la struttura dei cammini più brevi nella rete.
- **Closeness centrality**: determina quanto un nodo in una rete è "vicino" a tutti gli altri nodi $C(u) = \frac{n-1}{\sum_{V=1}^{n-1} d(v,u)}$ dove $d(v,u)$ è la distanza del percorso più breve tra v e u, mentre $n-1$ è il numero di nodi raggiungibili da u.
  Nel primo grafo i nodi Gianluca Demartini, Kevin Roitero e Stefano Mizzaro sono i nodi che hanno la closeness centrality più alta, infatti, sono cruciali per la comunicazione e il flusso di informazioni nella rete. Mentre nel grafo espanso, Damiano Spina ed Eddy Maddalena hanno i valori più alti di closeness centrality, il che è sensato. Il fatto che i nodi con la closeness centrality più alta siano cambiati dopo l'espansione della rete suggerisce l'emergere di nuovi nodi influenti.
- **Page rank**: indica l'importanza di un nodo in base alla quantità e qualità delle sue connessioni. Nel primo grafo i nodi con il page rank più alto sono rispettivamente Damiano Spina e Gianlcua Demartini ciò indica che non solo sono collegati a diversi nodi, ma che questi nodi sono anche importanti nella rete. Mentre nel grafo espanso sono Damiano Spina e Eddy Maddalena ad avere i valori più alti, indicando un cambiamento nella dinamica della rete.
- **HITS**:
  - **Hubness**: indica quanto un nodo può essere considerato come nodo "hub", ovvero un nodo che è molto connesso con gli altri nodi. Nel primo grafo i nodi con l'Hubness maggiore sono Kevin Roitero e Michael Soprano, mentre nel grafo espanso sono Damiano Spina e Eddy Maddalena.
  - **Authorities**: indica quanto un nodo è "autorevole", ovvero se è connesso da tanti nodi hub. Anche in questo caso nel primo grafo i nodi con valore di Authorities maggiore sono Kevin Roitero e Michael Soprano, mentre nel grafo espanso sono Damiano Spina e Eddy Maddalena

  Si può notare che i nodi hub siano anche autorities, in generale l'algoritmo HITS è stato sviluppato per grafi diretti, ma può essere applicato anche ai grafi indiretti come nel caso preso in analisi, infatti i concetti di autorities e di nodo hub possono essere interpretati come nodi che sono particolarmente centrali o rilevanti in termini di connettività con altri nodi.

3.7 Risultati Punto 7

Nell'ultima porzione del codice sono stati prodotti due file `.html` i quali garantiscono la possibilità di visualizzare i grafi in maniera interattiva: `first_graph` e `extended_graph`.

4 Conclusioni

Alla fine dello sviluppo del progetto, possono essere tratte le seguenti conclusioni dopo un'analisi svolta sui dati: l'analisi della rete ha evidenziato come il principio del preferential attachment abbia impattato significativamente sulla struttura della rete, promuovendo la formazione di connessioni preferenziali tra nodi già molto connessi, infatti il numero di nodi più connessi nel grafo espanso è aumentato.
Mentre per quanto riguarda la presenza di cluster e la tendenza dei nodi a formare dei triangoli nel grafo espanso, indicano una maggiore coesione e connettività tra gli autori, il che è coerente con l'aspettativa data dall'algoritmo del preferential attachment.
L'incremento delle connessioni tra nodi ha ridotto la distanza media tra di essi, e questo fenomeno è emerso attraverso la formazione di cluster più densi. In quanto la diminuzione osservata nella betweenness centrality potrebbe essere interpretata come il risultato della ridistribuzione del flusso di informazioni e la modifica della struttura dei cammini più brevi nella rete.
Per finire, i coefficienti di small-world indicano che il grafo espanso segue una struttura più simile a un grafo casuale, suggerendo che il preferential attachment contribuisce a una maggiore efficienza nella connettività.

In sintesi, il preferential attachment ha contribuito a modellare la rete in una forma più efficiente e coesa, l'analisi dei due grafi rivela che nel primo grafo i nodi tendono a formare cluster, con nodi centrali come Michael Soprano e Kevin Roitero. Nel grafo espanso, influenzato dal preferential attachment, emergono più nodi centrali, mantenendo comunque un raggio di 2. La distanza media si riduce, indicando percorsi più brevi. La transitività aumenta, suggerendo la formazione di cluster più densi, i coefficienti di small-world indicano una maggiore somiglianza a un modello small-world nel grafo espanso. Mentre nelle ultime porzioni di analisi si osserva una variazione nei nodi con maggiore centralità. Nel grafo espanso, la struttura della rete cambia, con una riduzione nella betweenness centrality. Closeness centrality e page rank mostrano variazioni, indicando cambiamenti nell'importanza dei nodi, in fine l'HITS rivela una redistribuzione dei ruoli di hub e authorities tra i due grafi, sottolineando l'effetto dell'espansione della rete mediante preferential attachment.