

Dipartimento di Scienze Aziendali - Management & Innovation Systems  
Corso di Laurea in Data Science & Gestione dell'Innovazione

## Il Gabibbo: una figura controversa

Come il Gabibbo ha creato un esercito di neofascisti nell'Italia del  
21esimo secolo

INFORMATION DISORDER AND GLOBAL SECURITY

# Echoes of the right: La genesi delle community altright su Telegram

Come i network digitali hanno forgiato l'altright su Telegram

by

Denise Brancaccio - MAT. 0222800163  
Lucia Brando - MAT. 0222800162  
Bruno Maria Di Maio - MAT. 0222800149

**Instructor:** G. Fenza  
**Teaching Assistant:** D. Cavalieri  
**Project Duration:** Month, Year – Month, Year  
**Faculty:** Data Science & Gestione dell’Innovazione



## Prefazione

*Il mondo delle community online ha assunto un ruolo sempre più centrale nella formazione delle opinioni, nella condivisione di informazioni e nella costruzione di identità collettive. Con la crescente diffusione di notizie false e la polarizzazione delle opinioni, è cruciale comprendere le dinamiche di queste community e il loro rapporto con la qualità delle fonti informative.*

*Questo lavoro si propone di esplorare il livello di apertura delle community su Telegram e il legame tra tale apertura e l'attendibilità dei siti di notizie condivisi al loro interno.*

*L'obiettivo è non solo mappare le connessioni tra le community e le fonti, ma anche fornire uno strumento di analisi visiva che evidenzi le dinamiche di interazione attraverso l'utilizzo di grafi generati con Gephi.*

*Questo studio si inserisce in un contesto di ricerca multidisciplinare che combina data science, analisi delle reti sociali e valutazione delle fonti informative, offrendo nuove prospettive sulla comprensione dell'ecosistema informativo digitale.*



# Indice

<b>1 Introduzione</b>	<b>4</b>
1.1 Obiettivo principale . . . . .	4
<b>2 Metodi e strumenti di lavoro</b>	<b>5</b>
2.1 Glossario . . . . .	5
2.2 Telegram . . . . .	7
2.3 NewsGuard . . . . .	8
2.4 Gephi . . . . .	8
2.5 Open Measures . . . . .	9
<b>3 Sviluppo del lavoro</b>	<b>10</b>
3.1 Raccolta e pre-analisi dei dati . . . . .	10
3.1.1 Utilizzo di Open Measures . . . . .	10
3.1.2 Raccolta dei dati . . . . .	11
3.1.3 Pre-analisi dei dati . . . . .	11
3.1.4 Esempio di righe da includere nella documentazione del file node.csv: . . . . .	12
3.1.5 Esempio di righe da includere nella documentazione del file archs.csv: . . . . .	12
3.1.6 Verifica d'integrità . . . . .	12
3.1.7 Strutturazione dei file per Gephi . . . . .	12
3.1.8 Valutazione dei dati . . . . .	13
3.1.9 Estrazione ed analisi dei topic . . . . .	14
<b>4 Grafi</b>	<b>14</b>
4.1 Creazione dei grafi . . . . .	14
4.1.1 Grafo orientato . . . . .	14
4.1.2 Grafo non orientato . . . . .	15
4.2 Analisi dei grafi . . . . .	15
4.2.1 Analisi del grafo orientato . . . . .	15
4.2.2 Analisi del grafo non orientato . . . . .	20
4.3 Analisi dei risultati . . . . .	20
<b>5 Clustering</b>	<b>20</b>
<b>6 Conclusione e valutazione finale del team</b>	<b>20</b>
6.1 Sintesi dei risultati . . . . .	20

# 1 Introduzione

Negli ultimi anni, la diffusione di informazioni sui social media e sulle piattaforme di messaggistica istantanea ha radicalmente trasformato il modo in cui le persone accedono alle notizie.

**Telegram**, in particolare, è emerso come uno degli strumenti più utilizzati per la condivisione di contenuti informativi, grazie alla sua flessibilità e alla capacità di ospitare gruppi e canali con migliaia di membri. Tuttavia, questa libertà d'uso si accompagna a rischi significativi, tra cui la proliferazione di notizie false e l'amplificazione di narrazioni manipolatorie. Le **community online** rappresentano un microcosmo di dinamiche sociali e informative, dove l'apertura – intesa come la capacità di interagire con fonti e utenti esterni – gioca un ruolo cruciale nel determinare la qualità e l'affidabilità delle informazioni condivise.

*Questo studio si propone di investigare il rapporto tra il livello di apertura delle community e l'attendibilità delle fonti giornalistiche che vi circolano, offrendo una panoramica approfondita delle interazioni tra utenti, contenuti e fonti informative.*

## 1.1 Obiettivo principale

L'obiettivo principale di questo lavoro è analizzare il livello di apertura delle community presenti su Telegram e correlare tale apertura all'affidabilità dei siti di notizie condivisi.

Attraverso l'utilizzo di dati raccolti direttamente da **Telegram**, incrociati con valutazioni fornite da **NewsGuard**, si mira a costruire un grafo rappresentativo delle connessioni e delle interazioni tra community e fonti informative.

Il grafo, realizzato tramite **Gephi**, consente una visualizzazione chiara e intuitiva delle dinamiche emerse, facilitando l'identificazione di pattern significativi e relazioni critiche.

## 2 Metodi e strumenti di lavoro

Per condurre l'analisi, è stato sviluppato uno **script Python** progettato per raccogliere dati da Telegram.

Questo script ha permesso di filtrare i messaggi contenenti link a siti di notizie.

I dati raccolti sono stati elaborati confrontandoli con i punteggi di **NewsGuard**, generando così un file .csv che includeva nodi (le community) e archi (le connessioni tra essi).

Il file è stato poi importato in **Gephi** per creare un grafo che rappresentasse visivamente le dinamiche informative.

### 2.1 Glossario

Di seguito si riportano i termini chiave utilizzati nel contesto della comunicazione e della diffusione delle informazioni.

Sono stati utilizzati termini che identificano contenuti di attualità e aggiornamenti di rilevante importanza:

- News
- Breaking
- Alert
- Headline
- Update

Termini che si riferiscono a diverse tipologie di documenti informativi e giornalistici:

- Report
- Article
- Scoop
- Flash

Termini che indicano fenomeni informativi trasmessi in tempo reale o caratterizzati da una rapida diffusione:

- Live
- Trending
- Viral

Termini che descrivono situazioni di urgenza o emergenza:

- Emergency
- Incident
- Accident
- Disaster
- Crisis

Sono stati utilizzati, inoltre, temi di rilevanza sociale e politica:

- Conflict
- War
- Pandemic
- Health
- Technology
- Economy
- Politics
- Elections
- Debate
- Protest

Ma anche termini che delineano strumenti, processi e dinamiche legati alla raccolta, elaborazione e diffusione dell'informazione:

- Survey
- Poll
- Results
- Statistics
- Scandal
- Investigation
- Inquiry
- Revelation
- Discovery
- Press conference
- On Air

## 2.2 Telegram



Telegram è una piattaforma di messaggistica molto utilizzata per la creazione di gruppi e canali tematici in cui gli utenti possono condividere e discutere contenuti.

A differenza di altre piattaforme, Telegram permette la creazione di canali pubblici e gruppi con un numero potenzialmente illimitato di membri, favorendo una rapida diffusione delle informazioni.

L'architettura di Telegram, infatti, consente agli utenti di condividere link a siti web, creando un ecosistema in cui notizie, articoli e contenuti virali possono diffondersi rapidamente. Questo lo rende un campo di studio ideale per analizzare la diffusione delle notizie, soprattutto in relazione alla qualità delle informazioni condivise.

Inoltre, la disponibilità di API permette di raccogliere dati in modo automatizzato e strutturato, semplificando l'analisi dei messaggi e dei contenuti condivisi all'interno dei gruppi e canali.

*Per il nostro studio, Telegram è stato scelto come piattaforma principale per monitorare e raccogliere i dati, in quanto fornisce un ampio spettro di community diverse e informazioni su un vasto numero di argomenti, spesso trattati in modo non filtrato e senza verifica.*

## 2.3 NewsGuard



NewsGuard è uno strumento che fornisce valutazioni affidabili sui siti web di notizie, assegnando loro un punteggio in base a criteri come la trasparenza, la correttezza e l'imparzialità dei contenuti.

Ogni sito analizzato viene esaminato da un team di giornalisti professionisti, che valuta i parametri di qualità delle notizie pubblicate e fornisce una valutazione che aiuta gli utenti a discernere tra fonti affidabili e quelle che potrebbero diffondere disinformazione.

L'integrazione di NewsGuard nel nostro studio è stata essenziale, in quanto ha permesso di incrociare i link raccolti da Telegram con una valutazione professionale e oggettiva della loro affidabilità.

*Grazie a questo strumento, è stato possibile determinare se le fonti di notizie condivise all'interno delle community fossero credibili o se provenissero da siti tendenti alla disinformazione. L'utilizzo di NewsGuard ha aggiunto un ulteriore livello di analisi alla nostra ricerca, consentendo di classificare i dati in modo oggettivo e basato su criteri di qualità giornalistica.*

## 2.4 Gephi



Gephi è un software open-source utilizzato per l'analisi e la visualizzazione di grafi e reti complesse.

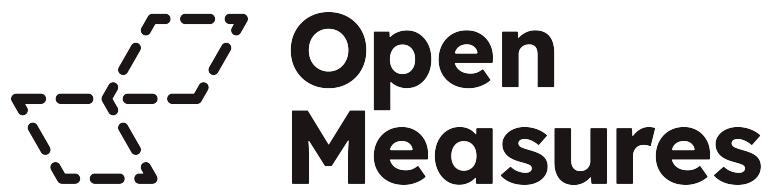
È particolarmente utile per studiare le relazioni tra entità attraverso la rappresentazione visiva di nodi e archi, dove ogni nodo rappresenta un'entità (in questo caso, una community), mentre gli archi rappresentano le connessioni tra di esse.

In questo studio, Gephi è stato utilizzato per visualizzare le connessioni tra i diversi nodi, permettendo una rappresentazione grafica che mostra la struttura e l'interconnessione di queste reti.

L'importazione dei dati raccolti e strutturati in formato CSV ha consentito di generare un grafo interattivo che rende facilmente visibili le relazioni tra le community e le fonti di notizie, offrendo al contempo una panoramica immediata delle dinamiche di diffusione delle informazioni.

*Grazie alla sua capacità di gestire grandi quantità di dati e generare visualizzazioni dinamiche e interattive, Gephi ha svolto un ruolo cruciale nell'interpretazione e nella presentazione dei risultati, consentendo di osservare pattern, cluster e altre caratteristiche significative nelle connessioni tra le community e i siti di notizie.*

## 2.5 Open Measures



Open Measures è una piattaforma avanzata progettata per supportare l'analisi e il monitoraggio delle interazioni sui social media, con un focus particolare sui contenuti generati dagli utenti.

La piattaforma offre API flessibili e potenti, che consentono di raccogliere dati strutturati su vasta scala.

Questi dati includono messaggi, metadati (ad esempio, autore, data e ora), contenuti multi-mediali, e informazioni sulle relazioni tra utenti o entità.

Le API di Open Measures sono progettate per essere altamente configurabili, permettendo di personalizzare le query attraverso parametri come parole chiave, intervalli temporali, e piattaforme social specifiche. Ciò consente agli utenti di accedere a dataset precisi e pertinenti per le loro analisi.

Ad esempio, è possibile eseguire ricerche avanzate per identificare discussioni che menzionano URL, frasi chiave o hashtag, oppure per analizzare contenuti pubblicati su piattaforme come Telegram, Twitter o Reddit.

Grazie alla sua architettura scalabile, Open Measures è particolarmente utile per progetti accademici, aziendali o governativi che richiedono l'analisi di grandi quantità di dati social.

Le sue applicazioni spaziano dal monitoraggio della disinformazione e dell'hate speech, all'analisi dei trend, fino agli studi sull'engagement e sulla polarizzazione delle opinioni.

Inoltre, la possibilità di esportare i dati in formati standard, come CSV, facilita l'integrazione con strumenti di analisi esterni come Python, R o software di visualizzazione come Gephi.

### 3 Sviluppo del lavoro

Il presente lavoro è articolato in diverse sezioni che riflettono il percorso di analisi intrapreso. Il seguente capitolo descriverà in dettaglio la metodologia adottata, illustrando le tecniche di raccolta, filtraggio e analisi dei dati.

Il successivo capitolo presenterà i risultati ottenuti, con particolare attenzione alla mappatura delle community e alla visualizzazione dei grafi.

Infine, verranno discussi i risultati evidenziando le implicazioni pratiche e teoriche, oltre a considerare i limiti dello studio e le potenziali direzioni future.

#### 3.1 Raccolta e pre-analisi dei dati

La raccolta e la preparazione dei dati costituiscono la base fondamentale di questo studio, che mira ad analizzare il comportamento delle community su Telegram.

In questa fase, è stata utilizzata una combinazione di strumenti di programmazione e dataset per strutturare le informazioni raccolte, filtrare i dati rilevanti e prepararli per l'analisi dei grafi.

##### 3.1.1 Utilizzo di Open Measures

Per questo progetto è risultato fondamentale l'utilizzo dell'API di Open Measures. Grazie a quest'ultima, infatti, è stato possibile accedere a grandi quantità di dati strutturati relativi a Telegram, includendo metadati come contenuti dei messaggi, timestamp ed informazioni sugli autori.

Per configurare il sistema, è necessario:

- Ottener un token di autorizzazione (salvato nel file `open-measures-key.txt`).
- Definire i parametri di ricerca attraverso file JSON (`parametri/required_columns.json`, `parametri/dtypes.json`, e `parametri/terms.json`), che specificano le colonne richieste, i tipi di dati e i termini di ricerca.

Un esempio di chiamata API con la funzione `fetch_results` mostra la struttura del processo di raccolta:

```
1 df = fetch_results(  
2     term="disinformazione",  
3     social="telegram",  
4     start_date="2024-01-01",  
5     end_date="2024-01-31"  
6 )
```

La funzione genera query dinamiche per l'API, suddivide automaticamente gli intervalli temporali in caso di limiti di risultati, e converte le risposte in un DataFrame pandas.

Per esempio, una query per il termine “*disinformazione*” su Telegram ha restituito dati relativi a messaggi contenenti URL (`message:http OR message:https`) durante l'intero mese di gennaio 2024.

Infine, i risultati sono stati salvati in file CSV per ogni termine analizzato, come mostrato nel seguente frammento di codice:

```
1     output_file = "csv/telegram_disinformazione_2024.csv"
2     df.to_csv(output_file, index=False)
```

Questa metodologia ha garantito la scalabilità del processo di raccolta e la creazione di una base dati robusta per le successive fasi di analisi.

### 3.1.2 Raccolta dei dati

Lo script Python utilizzato per la raccolta dati permette di ottenere un file.csv contenente informazioni sui messaggi scambiati su Telegram, ponendo attenzione solamente su quelli che contenevano un link ad un sito di notizie.

I dati includono dettagli sui canali, gli utenti e le interazioni effettuate da essi.

Lo script ha esaminato ogni messaggio nel dataset identificando e memorizzando:

- gli **ID univoci** degli utenti che hanno interagito con i canali
- I **canali** con cui gli utenti hanno avuto interazioni, rappresentando queste informazioni come nodi nel grafo finale.

```
1     for index, replies in enumerate(df["replies"]):
2         if isinstance(replies, str):
3             replies = ast.literal_eval(replies)
4         if isinstance(replies, dict) and replies["recent_repliers"]:
5             for user, _ in enumerate(replies["recent_repliers"]):
6                 if "user_id" in replies["recent_repliers"][user]:
7                     if replies["recent_repliers"][user]["user_id"]:
8                         if not user_id in dizionario:
9                             dizionario[replies["recent_repliers"][
10                                user]["user_id"]] = ("User", [df["channelusername"]][index])
11                         else:
12                             dizionario[replies["recent_repliers"][
13                                user]["user_id"]][1].append(df["channelusername"])[index])
```

### 3.1.3 Pre-analisi dei dati

Una volta raccolte le informazioni, si è passati alla scrittura dei dati in due file.csv distinti:

- **nodes.csv**: contiene l'elenco di nodi della rete, dove i nodi possono essere utenti o canali.
- **archs.csv**: descrive gli archi della rete, rappresentando le interazioni tra utenti e canali.

### 3.1.4 Esempio di righe da includere nella documentazione del file node.csv:

ID	Type
12345678	User
channel_1	Channel

### 3.1.5 Esempio di righe da includere nella documentazione del file archs.csv:

Source	Target	Another Target
12345678	channel_1	Reply

### 3.1.6 Verifica d'integrità

La pre-analisi dei dati si concentra sulla verifica della loro integrità e sulla loro trasformazione in formati compatibili con le fasi successive.

Dopo aver caricato i file CSV, è stata eseguita un'analisi esplorativa per identificare eventuali valori nulli o incongruenze:

```
1 df = pd.read_csv("csv/telegram_disinformazione_2024.csv")
2 print(df.info())
3 print(df.describe())
```

Un altro passaggio cruciale è stato la conversione delle colonne in tipi di dati uniformi utilizzando un file JSON contenente i tipi previsti (**dtypes.json**).

Questa conversione ha evitato errori nella manipolazione successiva dei dati:

```
1 with open("parametri/dtypes.json", "r") as f:
2     dtypes = json.load(f)
3     df = df.astype(dtypes)
```

Infine, le colonne non essenziali sono state eliminate per migliorare l'efficienza del sistema, mantenendo solo quelle richieste, come specificato nel file **required\_columns.json**.

### 3.1.7 Strutturazione dei file per Gephi

Per la visualizzazione dei dati e l'analisi delle reti, i dati raccolti sono stati convertiti in formati compatibili con Gephi.

Sono stati generati due file CSV:

- **File dei nodi:** specifica il tipo di ogni entità (utente o canale).
- **File degli archi:** specifica la relazione tra i nodi (ad esempio, un utente che risponde a un canale).

Lo script `gephi.py` ha automatizzato questa trasformazione.

Per esempio, i nodi sono stati creati come segue:

```
1 with open("nodes.csv", mode="w", newline="", encoding="utf-8"):
2     ) as file:
3         writer = csv.writer(file)
```

```

3     writer.writerow(["id", "type"])
4     for user_id, (user_type, channels) in dizionario.items():
5         writer.writerow([user_id, user_type])
6         for channel in channels:
7             writer.writerow([channel, "Channel"])

```

Gli archi, invece, sono stati strutturati per rappresentare la relazione tra utenti e canali:

```

1     with open("archs.csv", mode="w", newline="", encoding="utf-8"):
2         ) as file:
3             writer = csv.writer(file)
4             writer.writerow(["source", "target", "type"])
5             for user_id, (_, channels) in dizionario.items():
6                 for channel in channels:
7                     writer.writerow([user_id, channel, "Reply"])

```

Questa strutturazione ha permesso di importare facilmente i dati in Gephi e analizzarli visivamente.

### 3.1.8 Valutazione dei dati

Una valutazione preliminare dei dati è stata condotta per garantire la loro qualità e idoneità.

Durante questa fase, sono stati utilizzati script per calcolare metriche chiave come la distribuzione delle interazioni per canale e l'identificazione di utenti con un elevato livello di attività:

```

1     channel_counts = df["channelusername"].value_counts()
2     print("Distribuzione delle interazioni per canale:\n",
3           channel_counts.head())

```

Sono stati anche identificati utenti connessi a più canali, indicativi di potenziali influencer:

```

1     for user_id, (_, channels) in dizionario.items():
2         if len(channels) > 5:
3             print(f"L'utente {user_id} ha interagito con {len(channels)} canali.")

```

Questi risultati sono stati utilizzati per affinare ulteriormente il dataset, eliminando eventuali anomalie e preparando i dati per l'analisi di rete in Gephi.

### 3.1.9 Estrazione ed analisi dei topic

## 4 Grafi

### 4.1 Creazione dei grafi

#### 4.1.1 Grafo orientato

Il grafo riportato rappresenta una rete di utenti e comunità presenti su Telegram, modellata attraverso una struttura complessa di nodi e archi.

Questa rappresentazione è stata ottenuta utilizzando il software Gephi, attraverso il caricamento dei dataset **node.csv** e **archs.csv**, contenenti rispettivamente le informazioni relative ai nodi (utenti o comunità) e agli archi (connessioni o interazioni tra i nodi).

I nodi, visualizzati con colori distinti, identificano cluster di utenti che condividono connessioni più dense, suggerendo l'esistenza di comunità tematiche o gruppi coesi.

Gli archi, che collegano i nodi, rappresentano le interazioni o le connessioni dirette tra i diversi utenti o gruppi, evidenziando la struttura globale della rete.

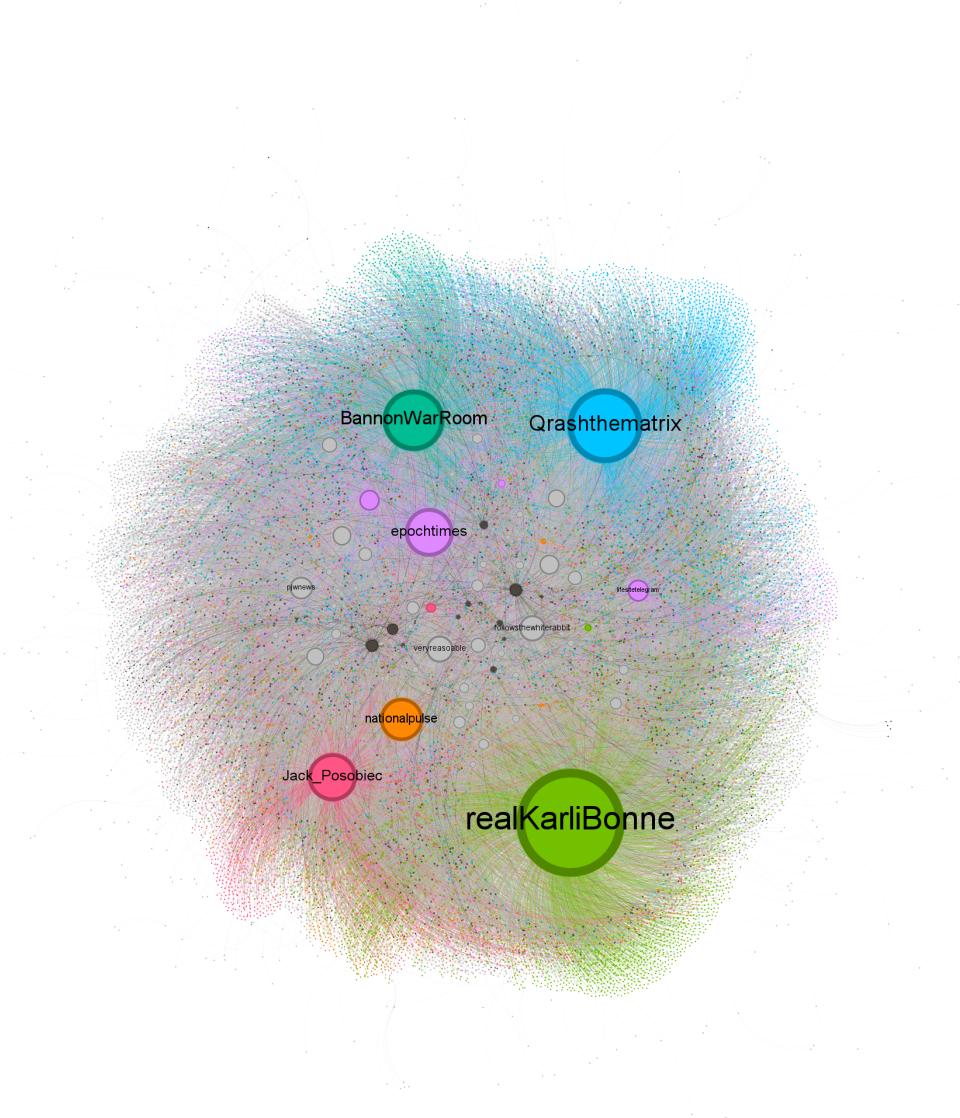
La distribuzione spaziale del grafo evidenzia una centralità maggiore in alcune aree, dove i nodi risultano più grandi e prominenti.

Questi possono essere interpretati come utenti o gruppi particolarmente influenti all'interno della rete, il cui ruolo potrebbe essere cruciale nella diffusione di informazioni o nel consolidamento di opinioni.

Al contrario, i nodi periferici, distanziati dal nucleo centrale, possono rappresentare utenti marginali o comunità meno integrate, contribuendo comunque alla diversità della rete complessiva.

La colorazione differenziale dei cluster permette una chiara identificazione delle comunità, facilitando l'analisi di fenomeni come la polarizzazione, l'influenza e la diffusione di informazioni all'interno del network.

L'osservazione visiva del grafo mostra come le comunità tendano a formare strutture relativamente compatte, con alcune interconnessioni tra gruppi distinti, suggerendo possibili punti di scambio informativo o di sovrapposizione tematica.



#### 4.1.2 Grafo non orientato

### 4.2 Analisi dei grafi

#### 4.2.1 Analisi del grafo orientato

Il seguente codice implementa un'analisi strutturale sul grafo orientato e esportato in formato **.graphml**.

Esso utilizza librerie Python come **NetworkX** per calcoli di metriche topologiche avanzate e **Pandas** per l'organizzazione e l'esportazione dei risultati.

Il grafo viene analizzato attraverso misure di centralità e metriche di rete.

Di seguito è riportata una spiegazione dettagliata:

#### ○ Caricamento del grafo:

Il grafo viene caricato utilizzando il metodo **nx.read\_graphml**, che importa dati

strutturati in formato **.graphml**. Una volta caricato, vengono stampate informazioni generali sulla rete, come il numero di nodi e di archi:

```
1     print("Grafo caricato con successo. Numero di nodi:",  
      graph.number_of_nodes(), "Numero di archi:", graph.  
      number_of_edges())
```

Questo consente una verifica preliminare per assicurarsi che i dati siano stati importati correttamente.

- **Filtraggio dei nodi "channel":** Il codice filtra i nodi con un **attributo "type"** pari a **"Channel"**, utilizzando una lista comprensione:

```
1     channels = [node for node in graph.nodes if graph.nodes[  
      node].get("type") == "Channel"]
```

Questa operazione seleziona un sottoinsieme specifico del grafo, così da poter focalizzare l'analisi su canali Telegram piuttosto che su altri tipi di nodi.

Il numero totale di nodi selezionati viene stampato per trasparenza.

- **Calcolo delle metriche di centralità:**

- **Degree:**

Il grado rappresenta il numero di connessioni di un nodo ed è calcolato senza considerare eventuali pesi sugli archi:

```
1     degree_dict = dict(graph.degree(channels))
```

Questa metrica è utile per identificare nodi altamente connessi all'interno della rete.

- **Betweenness Centrality:**

Utilizzando l'algoritmo ottimizzato di Brandes, il codice misura quanto un nodo si trovi su percorsi più brevi tra coppie di altri nodi.

Questo indica l'importanza del nodo come intermediario:

```
1     betweenness_dict = nx.betweenness_centrality(  
      graph, normalized=True, endpoints=False)
```

L'opzione **normalized=True** assicura che i valori siano scalati rispetto alla dimensione del grafo.

- **Closeness Centrality:**

Questa metrica misura quanto un nodo sia vicino a tutti gli altri nodi della rete, in termini di distanza geodetica:

```
1     closeness_dict = nx.closeness_centrality(graph)
```

- **Authority Score:**

Se il grafo è orientato, viene calcolata l'authority di ciascun nodo utilizzando

l'algoritmo HITS, che assegna punteggi in base alla qualità delle connessioni in entrata:

```
1     _, authority_dict = nx.hits(graph, normalized=True)
```

Nel caso di grafi non orientati, l'authority viene impostata a zero, poiché Gephi non supporta il calcolo di questa metrica per grafi non diretti.

#### – **Eigenvector Centrality:**

Questa metrica valuta l'influenza di un nodo in base all'importanza dei suoi vicini, calcolata iterativamente:

```
1     eigenvector_dict = nx.eigenvector_centrality(
2         graph, max_iter=1000, tol=1e-6)
```

Parametri come **max\_iter** e **tol** sono scelti per garantire la convergenza.

### ○ Creazione dei risultati:

I risultati delle metriche vengono raccolti in una lista di dizionari, dove ciascun dizionario rappresenta un nodo e le relative metriche calcolate:

```
1     results = []
2     for node in channels:
3         results.append({
4             "Node": node,
5             "Degree": degree_dict.get(node, 0),
6             "Betweenness": betweenness_dict.get(node, 0),
7             "Closeness": closeness_dict.get(node, 0),
8             "Authority": authority_dict.get(node, 0),
9             "Eigenvector": eigenvector_dict.get(node, 0),
10            })
```

### ○ Esportazione risultati:

I risultati vengono convertiti in un DataFrame di Pandas, una struttura dati tabellare, e salvati in **formato .csv**:

```
1     df = pd.DataFrame(results)
2     output_file = "channel_metrics_gephi.csv"
3     df.to_csv(output_file, index=False)
```

La fase successiva dell'analisi del grafi prevede che i dati derivati vengano elaborati ulteriormente per calcolare una nuova metrica chiamata "**tasso di apertura**" (**Openness**).

Questo processo consente di arricchire ulteriormente le informazioni estratte, aggiungendo una dimensione sintetica che combina diverse metriche di centralità per fornire una visione complessiva dell'importanza e della posizione strategica dei nodi della rete.

Lo script si basa sui due file: il primo, **statpython.csv**, contiene i dati di base del grafo già processati e arricchiti con metriche di centralità; il secondo, **statpython\_with\_openness.csv**, viene generato dallo script ed include la nuova colonna "Openness" accanto alle metriche

originali, ampliando così le possibilità di analisi successive.

Il file **statpython.csv** rappresenta una tabella con informazioni dettagliate per ogni nodo del grafo. Le colonne principali includono:

- **Node:** identificativo univoco del nodo nella rete. Questo attributo consente di distinguere ciascun elemento analizzato, garantendo l'integrità dei collegamenti tra i dati originali e quelli derivati.
- **Degree:** numero di connessioni dirette che un nodo ha nella rete, una misura semplice ma fondamentale della sua interconnessione.
- **Betweenness:** misura dell'importanza di un nodo come intermediario nei percorsi brevi tra altri nodi, evidenziando il suo ruolo nel facilitare il flusso di informazioni.
- **Closeness:** rappresenta la vicinanza di un nodo a tutti gli altri nodi della rete, calcolata come l'inverso della somma delle distanze, una metrica utile per identificare nodi centrali.
- **Authority:** valore che indica l'autorevolezza del nodo basato sull'algoritmo HITS, particolarmente rilevante in grafi diretti dove l'influenza reciproca tra nodi gioca un ruolo cruciale.
- **Eigenvector:** centralità che misura l'influenza del nodo in base all'importanza dei nodi a cui è connesso, una metrica iterativa che considera il contesto globale della rete

Un esempio delle prime cinque righe di questo file è il seguente:

```
1     Node , Degree , Betweenness , Closeness , Authority , Eigenvector
2     1 , 10 , 0.02 , 0.4 , 0.3 , 0.8
3     2 , 5 , 0.01 , 0.35 , 0.2 , 0.7
4     3 , 8 , 0.03 , 0.38 , 0.25 , 0.75
5     4 , 6 , 0.015 , 0.37 , 0.22 , 0.72
6     5 , 7 , 0.02 , 0.36 , 0.24 , 0.73
```

Questa struttura garantisce la chiarezza e la coerenza necessarie per operazioni successive di analisi e calcolo.

Lo script inizia caricando i dati da **statpython.csv** utilizzando la **libreria pandas**, un potente strumento per la manipolazione di dati strutturati.

Dopo aver caricato i dati, verifica che le colonne necessarie siano presenti, garantendo che il dataset sia completo e conforme ai requisiti dell'analisi:

```
1     file_path = 'statpython.csv'
2     df = pd.read_csv(file_path)
3     required_columns = ["Node", "Degree", "Betweenness", "
4         Closeness", "Authority", "Eigenvector"]
5     if not all(col in df.columns for col in required_columns):
6         raise ValueError(f"Il file CSV deve contenere le seguenti
7             colonne: {', '.join(required_columns)}")
```

Se una delle colonne mancasse, lo script interromperebbe l'esecuzione con un messaggio di errore chiaro, assicurando un controllo di qualità sui dati in ingresso.

Per rendere comparabili le diverse metriche, lo script utilizza il metodo **MinMaxScaler** di **sklearn** per scalare i valori nel range [0,1].

Questa normalizzazione è essenziale per eliminare eventuali disparità dovute a scale diverse nelle metriche originali e garantire che ogni metrica contribuisca equamente al calcolo successivo:

```
1     metrics = ["Degree", "Betweenness", "Closeness", "Authority",
2                  "Eigenvector"]
3     scaler = MinMaxScaler()
4     normalized_metrics = scaler.fit_transform(df[metrics])
```

Il risultato è una matrice in cui ogni valore rappresenta una versione scalata del corrispondente valore originale, uniformando l'importanza relativa di ciascuna metrica.

La metrica "**Openness**" viene calcolata come la media aritmetica delle metriche normalizzate per ogni nodo.

Questo approccio aggrega informazioni complesse in un unico valore, semplificando l'identificazione di nodi che mostrano una combinazione equilibrata di centralità e influenza:

```
1 df["Openness"] = normalized_metrics.mean(axis=1)
```

Il risultato è una nuova colonna aggiunta al dataset, che fornisce un'indicazione sintetica ma potente del ruolo globale del nodo nella rete analizzata.

Il file aggiornato, contenente la nuova colonna "Openness", viene salvato in un nuovo file CSV.

Questa operazione consente di conservare i dati elaborati in un formato facilmente condivisibile e compatibile con ulteriori analisi:

```
1 output_file_path = "statpython_with_openness.csv"
2 df.to_csv(output_file_path, index=False)
```

Le prime cinque righe del file generato appaiono così:

```
1 Node, Degree, Betweenness, Closeness, Authority, Eigenvector,
2   Openness
3   1, 10, 0.02, 0.4, 0.3, 0.8, 0.508
4   2, 5, 0.01, 0.35, 0.2, 0.7, 0.432
5   3, 8, 0.03, 0.38, 0.25, 0.75, 0.488
6   4, 6, 0.015, 0.37, 0.22, 0.72, 0.465
7   5, 7, 0.02, 0.36, 0.24, 0.73, 0.482
```

Questa tabella arricchita permette di ottenere informazioni preziose in modo rapido ed efficiente.

Questo approccio permette di integrare informazioni complesse in un formato più facilmente interpretabile, utile per identificare nodi particolarmente influenti o strategici nella rete.

Inoltre, la combinazione dei due file consente una pipeline analitica ben definita, in cui ogni fase aggiunge un livello di approfondimento, migliorando la comprensione della struttura e delle dinamiche della rete.

**4.2.2 Analisi del grafo non orientato**

**4.3 Analisi dei risultati**

**5 Clustering**

**6 Conclusione e valutazione finale del team**

**6.1 Sintesi dei risultati**