UNIVERSITÀ DELLA CALABRIA
DIPARTIMENTO DI
INGEGNERIA INFORMATICA,
MODELLISTICA, ELETTRONICA
E SISTEMISTICA
DIMES

Corso di Laurea Ingegneria Informatica:

Artificial Intelligence & Machine Learning

Progetto Analisi di Social Network e Media

Community Detection e Influence Propagation in un Marketplace C2C: il caso Vinted

Professore:

Andrea Tagarelli

Candidato:

Silvio Emanuele Liparoti

Mat. 242663

ANNO ACCADEMICO 24/25

Indice

Capitolo 1

Introduzione e obiettivi del progetto

1.1 Contesto e Motivazione

Negli ultimi anni le piattaforme *consumer-to-consumer* (C2C) hanno assunto un ruolo centrale nei processi di scambio tra privati. Marketplace come Vinted, Wallapop o similari non sono semplici vetrine digitali, ma veri e propri ecosistemi sociali in cui utenti sconosciuti interagiscono, negoziano, costruiscono fiducia, valutano e vengono valutati.

Ogni transazione non rappresenta soltanto il passaggio di un oggetto, ma anche lo scambio di un giudizio che, esattamente come un arco in un grafo, crea un legame diretto tra due persone. Dal punto di vista dell'Analisi dei Social Network (SNA), tali piattaforme con feedback sono esempi di **reti sociali latenti**, ossia reti che non nascono esplicitamente come social network ma che emergono come tali dalle interazioni tra gli utenti.

La dinamica di feedback presente in Vinted introduce un meccanismo di **reputazione bidirezionale**: ogni transazione genera due valutazioni, una da Acquirente a Venditore e una da Venditore ad Acquirente. Questa caratteristica trasforma naturalmente il marketplace in un **grafo diretto e pesato**, dove:

- i **nodi** rappresentano gli utenti univoci coinvolti,
- gli **archi** sono le relazioni di fiducia generate dai feedback,

- il **peso dell'arco** corrisponde al rating numerico (1–5) assegnato durante la transazione.

La motivazione del progetto nasce dalla volontà di comprendere se, e in che modo, la struttura sociale implicita di un marketplace influenzi i fenomeni di reputazione e la diffusione dell'informazione. Questo lavoro esplora un caso reale e non simulato, contribuendo alla comprensione dei meccanismi di fiducia nei sistemi digitali e offrendo un framework replicabile per lo studio di piattaforme con interazioni utente-utente.Vinted è un caso di studio ideale: pur essendo una piattaforma relativamente chiusa (API non pubbliche), presenta un'elevata eterogeneità semantica tra gli utenti (collezionisti, venditori di moda, appassionati tech, ecc.), permettendo così di analizzare non solo gli aspetti strutturali del grafo, ma anche quelli *semantici*.

1.2 Obiettivi e Domanda di Ricerca

L'obiettivo generale del progetto è trasformare Vinted da semplice marketplace ad un **social network analizzabile**, ricostruendo l'intera rete delle transazioni e applicando i principali strumenti della Social Network Analysis. Queste prime analisi permettono di rispondere alla seguente domanda di ricerca:

> *In una rete di transazioni utente-utente, l'efficacia della diffusione della reputazione è determinata principalmente dalla coesione interna delle comunità (alta modularità) o dalla presenza di nodi ponte ad alta betweenness centrality?*

In altre parole: la fiducia si diffonde meglio **all'interno** delle community o **attraverso** di esse?

1.3 Sfide Tecniche e Scelte Progettuali

Una parte significativa del progetto è dedicata alla fase di acquisizione dati, resa complessa dal fatto che:

- Vinted non espone API pubbliche relative alle interazioni sociali,
- i Termini di Servizio vietano pratiche di scraping,
- la piattaforma integra meccanismi anti-bot.

Per aggirare tali limitazioni (senza violare le linee etiche del progetto) è stato adottato un approccio di **crawling controllato**, attraverso:

- utilizzo di un account dedicato;
- routing del traffico attraverso VPN (NordVPN);
- gestione manuale dei cookie di sessione;
- interrogazioni lente e distanziate (es. `time.sleep(15)`);
- utilizzo di una libreria Python non ufficiale (`vinted-api-wrapper`).

La pipeline di acquisizione è suddivisa in tre fasi principali:

1. **Tagging Semantico**: assegnazione automatica di un Main Tag e di un Detailed Tag a ciascun utente;
2. **Recupero delle Recensioni**: per ogni transazione vengono recuperati entrambi i rating (Acquirente → Venditore e Venditore → Acquirente).
3. **Costruzione del dataset**: costruzione del dataset a partire dai dati raccolti che servirà per la fase successiva di Analisi Social.

La pipeline dell'analisi social segue, invece, questi passi:

1. **Load e costruzione del grafo:** caricamento del dataset e successiva costruzione di un grafo orientato in cui i nodi rappresentano utenti ed includono i tag semantici, mentre gli archi rappresentano feedback direzionato.
2. **Metriche strutturali e similarity:** calcolo di statistiche globali (numero di nodi/archi, densità, reciprocità) e misure di similarità/topologia come il clustering medio e una stima di Jaccard per valutare l'overlap dei vicini.

3. **Calcolo delle centralità:** valutazione delle principali misure di importanza dei nodi — degree centrality (in/out), PageRank e betweenness centrality.

4. **Community detection:** applicazione dell'algoritmo di Louvain sul grafo non orientato per rilevare community e calcolare la modularità;

5. **Naming delle community:** assegnazione di etichette leggibili alle community (es. *MODA & LUSSO*, *TRADING CARD GAMES*) analizzando la distribuzione dei *main_tag* all'interno di ogni cluster;

6. **Influence Maximization (CELF) e confronto strategie:** esecuzione di una procedura greedy (CELF) per selezionare un set di seed massimizzanti la diffusione secondo il modello *Independent Cascade*; confronto sperimentale con baseline semplici (Top out-degree, Top PageRank) mediante simulazioni Monte Carlo.

7. **Analisi delle incongruenze nelle recensioni:** verifica delle coppie Acquirente–Venditore con rating discordanti per individuare possibili anomalie o bias nelle valutazioni;

8. **Esportazione e preparazione per Gephi:** arricchimento del grafo con gli attributi calcolati e salvataggio in formato .gexf per visualizzazione e post-processing in Gephi.

I parametri chiave usati in questa fase (valori configurabili nello script) includono il numero di campioni per l'approssimazione della betweenness, la probabilità di attivazione per il modello IC, il numero di simulazioni Monte Carlo per valutare lo spread e il numero di seed target per CELF. I risultati numerici prodotti sono sintetizzati nei file CSV di output e consentono sia analisi quantitative sia una visualizzazione qualitativa tramite Gephi.

Schema Riassuntivo della Pipeline

In ultima istanza, l'intero processo può essere riassunto in questo modo:

$$\text{Scraping} \rightarrow \text{Dataset} \rightarrow \text{Grafo} \rightarrow \text{SNA} \rightarrow \text{Interpretazione}$$

Capitolo 2

Metodologia di Modellazione e Acquisizione Dati

Questo capitolo descrive in dettaglio la pipeline di acquisizione dati progettata per costruire il dataset ASNM. La metodologia adottata ha dovuto fronteggiare le sfide imposte dalle contromisure anti-scraping della piattaforma Vinted, richiedendo un approccio stratificato che combina tecniche di crawling controllato, discovery semantica dei nodi e logiche di self-healing.

2.1 Recupero e integrazione del wrapper API

La base tecnologica del progetto è costituita da una libreria di *reverse engineering* dell'API di Vinted, reperita su GitHub[1]. Questa scelta si è resa necessaria data l'assenza di API pubbliche ufficiali per l'accesso ai dati sociali (feedback e relazioni utente).
Il setup dell'ambiente di sviluppo ha previsto:

1. Clonazione del repository e configurazione di un ambiente virtuale Python isolato.

2. Installazione delle dipendenze critiche per la gestione delle richieste HTTP (`requests`) e l'analisi dei dati (`pandas`, `networkx`).

[1]Wrapper API: Pawikoski, *vinted-api-wrapper*, `https://github.com/Pawikoski/vinted-api-wrapper`.

3. Patching manuale di alcune funzioni del wrapper per gestire correttamente i nuovi endpoint dell'API `v2` di Vinted.

2.2 Preparazione dell'ambiente di scraping e acquisizione cookie

Per garantire l'accesso a endpoint protetti (come quelli relativi ai feedback utente) e mitigare il rischio di ban, è stata implementata una strategia di isolamento del traffico:

1. **Isolamento IP:** Utilizzo di **NordVPN** per instradare il traffico attraverso nodi residenziali, evitando di esporre l'indirizzo IP istituzionale/-domestico.

2. **Gestione Sessione:** Utilizzo di un account Vinted secondario accessibile tramite browser in modalità incognito.

3. **Cookie Injection:** Estrazione manuale del cookie di sessione autenticato (`_vinted_fr_session`) tramite DevTools e sua iniezione nel client Python tramite la variabile `RAW_COOKIE_STRING`.

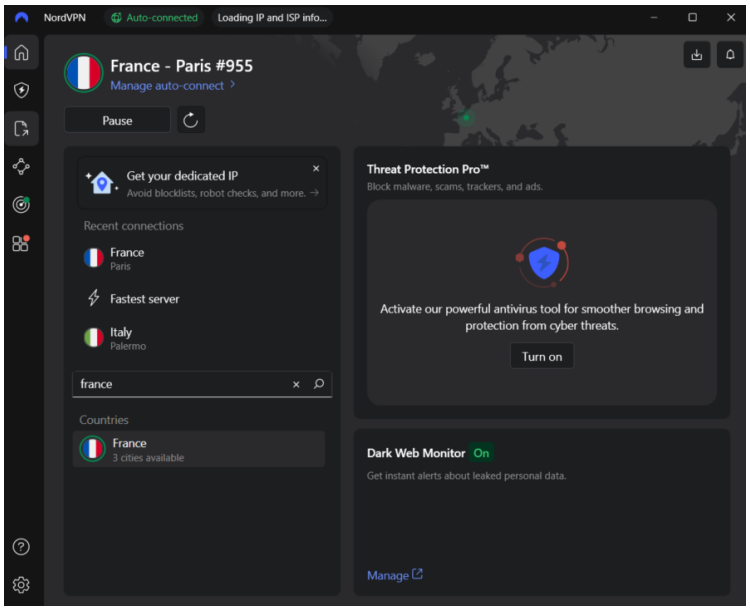

Figura 2.1: Connessione ad un server VPN per l'isolamento del traffico.

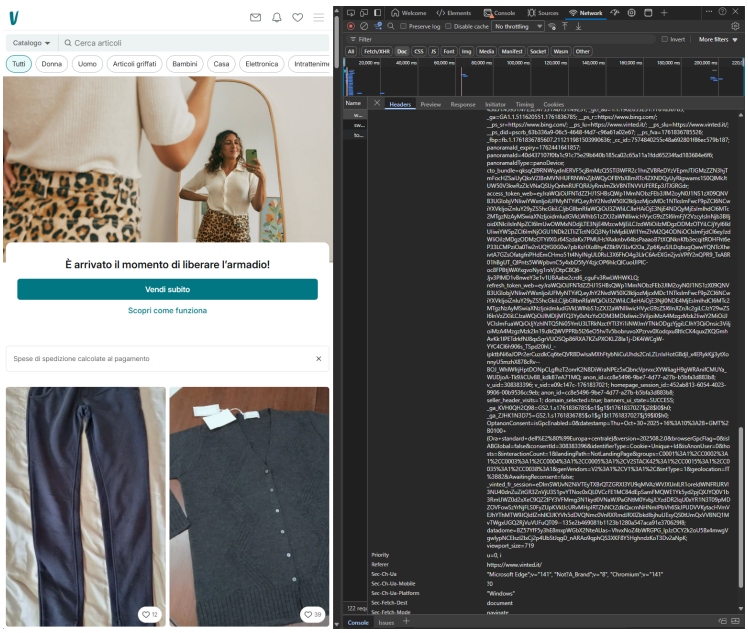

Figura 2.2: Acquisizione del cookie di sessione autenticato.

2.3 Identificazione e Selezione dei Nodi Seed

La fase successiva all'impostazione dell'ambiente di scraping ha riguardato l'individuazione dei nodi sorgente da cui avviare l'esplorazione della rete Vinted. Poiché l'obiettivo del progetto non era costruire un grafo uniforme o casuale, ma osservare la struttura relazionale emergente attorno a specifiche comunità tematiche, è stato necessario definire un insieme iniziale di utenti (*Seed Users*) rappresentativi delle diverse nicchie di mercato. La selezione dei Seed costituisce, di fatto, il primo passo di modellazione concettuale del dataset: scegliendo gli ingressi appropriati, si controlla l'orientamento dell'intera espansione della rete.

Per evitare un processo manuale lungo e soggettivo, la raccolta degli ID candidati è stata automatizzata interrogando l'endpoint del catalogo di Vinted. È stato definito un dizionario di **Query di Interesse** (`INTEREST_QUERIES`) progettato per coprire le macro-categorie oggetto di studio.

La lista delle query utilizzate comprende termini specifici idonei a intercettare nicchie di mercato ben definite:

```
INTEREST_QUERIES = [
    # Collezionismo e Nerd Culture
```

```
    "Funko Pop", "Pok mon", "Pok mon TCG", "Manga", "Yu-Gi-Oh
    ↪ !",

    # Gaming e Tech
    "Playstation", "Nintendo Switch", "Game Boy", "Apple",

    # Moda e Lusso
    "Gucci", "Prada", "Dior", "Nike", "Adidas",

    # Altro (Categorie di controllo)
    "Vinile", "Thun", "Ikea"
    # ... (lista completa nello script originale)
]
```

Listing 2.1: Esempio delle query di interesse per la selezione dei Seed

Successivamente lo script ha interrogato l'API per ciascun termine recuperando i venditori associati ai primi risultati rilevanti. La logica impiegata nella funzione `find_ids_from_raw_json` non si appoggia al wrapper, ma effettua una richiesta diretta al catalogo, così da accedere alla rappresentazione grezza dei dati. Ciò consente di minimizzare gli effetti delle astrazioni imposte dalla libreria e permette di sfruttare pienamente la semantica della ricerca interna alla piattaforma.

```
def find_ids_from_raw_json(query_text: str) -> List[int]:
    # Costruzione URL diretto all'API V2
    url = (
        "https://www.vinted.it/api/v2/catalog/items?"
        f"page=1&per_page=5&search_text={query_text.replace('
    ↪ ', '+')}"
    )

    try:
        # Esecuzione richiesta con rotazione User-Agent
        response = vinted.scraper.get(url, headers=getattr(
    ↪ vinted, "headers", {}), cookies=getattr(vinted, "cookies"
    ↪ , {}), proxies=getattr(vinted, "proxy", None))

        real_seed_ids = []
        if 'items' in raw_json_data and raw_json_data['items']:
```

```
14            print(f"Trovati {len(raw_json_data['items'])}
    ↪ prodotti. ID Venditori:")
15            for item in raw_json_data['items']:
16                user = item.get('user') or {}
17                vendor_id = user.get('id')
18                vendor_login = user.get('login', '<no-login>')
19                if vendor_id and vendor_id not in real_seed_ids
    ↪ :
20                    print(f"  - Login: {vendor_login}, ID: {
    ↪ vendor_id}")
21                    real_seed_ids.append(vendor_id)
22            return real_seed_ids
23        else:
24            print("Nessun prodotto trovato nel JSON grezzo.")
25            return []
26
27    except Exception as e:
28        print(f"ERRORE GRAVE nella ricerca grezza: {e}")
29        return []
```

Listing 2.2: Estrazione degli ID venditori tramite API del catalogo

L'estrazione automatica non è stata considerata sufficiente per definire la lista definitiva dei Seed. La presenza di una keyword nel titolo di un oggetto può infatti derivare tanto da un venditore specializzato quanto da un utente occasionale che pubblica un singolo articolo fuori categoria. Per evitare che tali elementi disturbassero la successiva analisi di comunità, ogni insieme di candidati estratti è stato sottoposto a una validazione umana rapida ma rigorosa, mirata a verificare coerenza semantica, attività recente e una quantità consistente di transazioni. La lista risultante – circa ottanta utenti – è stata quindi congelata all'interno dello script, garantendo riproducibilità e stabilità nelle successive fasi di crawling.

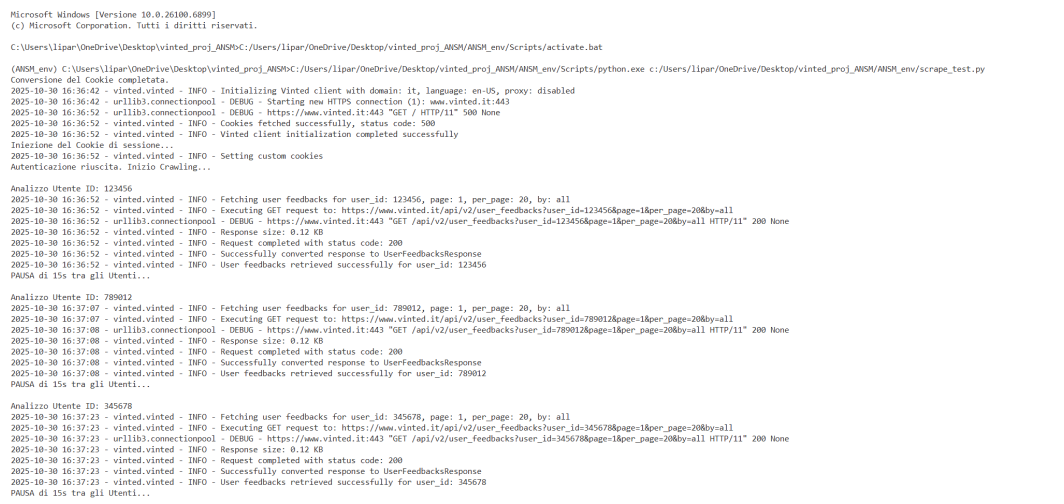

Figura 2.3: Log dell'estrazione automatizzata degli ID Seed tramite query semantiche.

2.4 Scraping dei Dati e Tagging Semantico

Una volta definito l'insieme dei Seed, la pipeline entra nella fase operativa più importante dell'intero progetto: la raccolta degli archi della rete e la categorizzazione semantica dei nodi. L'obiettivo non è soltanto estrarre informazioni sulle transazioni, ma costruire una rappresentazione coerente dell'ecosistema commerciale di Vinted, integrando struttura relazionale e informazione tematica. Per questo motivo, lo scraping e il tagging non sono processi separati: procedono in parallelo e si influenzano reciprocamente, permettendo di mappare la rete e, allo stesso tempo, descriverne le comunità interne.

2.4.1 Algoritmo di Tagging Semantico

Vinted non fornisce una categorizzazione esplicita dei venditori. L'identità commerciale degli utenti deve quindi essere inferita a partire dai segnali linguistici presenti nei titoli degli articoli che vendono o acquistano. L'idea centrale è che la ripetizione di certe keyword – legate, ad esempio, al collezionismo dei giochi di carte collezionabili (TCG), alla moda, ai fumetti o all'elettronica – permetta di individuare la nicchia commerciale di appartenenza dell'utente.
La funzione `assign_community_tag` implementa questo principio traducendolo in un processo strutturato di conteggio delle keyword. L'algoritmo mappa

ogni titolo in una o più categorie attraverso un dizionario lessicale; i conteggi risultanti guidano l'assegnazione di due etichette: 1) il *Main Tag*, che identifica la macro-categoria principale, e 2) il *Detailed Tag*, che ne specifica il grado di specializzazione.
Per stabilire se un utente sia un generalista, uno specialista o un nodo multi-categoria, la funzione utilizza quattro soglie di controllo:

- `MIN_ABS_FOR_DOMINANT` = 3: numero minimo di occorrenze richieste per riconoscere una nicchia dominante.
- `MULTICATEGORY_RATIO` = 0.4: rapporto minimo tra la seconda e la prima categoria per classificare un nodo come *Multi_category*.
- `MULTICATEGORY_MIN_ABS` = 2: numero minimo di occorrenze nella seconda categoria affinché sia considerata rilevante.
- `MIN_TOTAL_MATCHES` = 4: numero minimo di articoli analizzati per definire un profilo significativo per quella categoria.

Queste soglie, che a primo impatto possono risultare basse in termini assoluti se confrontate con l'inventario totale di un venditore professionale, sono state rigorosamente calibrate sulla finestra di osservazione dello scraper.Poiché per ovvie ragioni di efficienza l'acquisizione si limita allo storico recente delle transazioni (i feedback visibili nelle prime pagine), il rilevamento di 3 o 4 oggetti appartenenti a una nicchia specifica (es. *Lorcana*, *Gucci*) all'interno di un campione ridotto rappresenta una densità tematica molto elevata (spesso superiore al 20% dell'attività osservata). Inoltre, data l'alta specificità delle keyword utilizzate, la probabilità di "falsi positivi" è trascurabile: nel contesto C2C, la vendita ripetuta di articoli di nicchia non è quasi mai casuale, ma indica una precisa intenzionalità commerciale. L'adozione di soglie più elevate, tipiche del retail B2C, avrebbe comportato l'esclusione della "coda lunga" di piccoli collezionisti e utenti privati, che costituiscono invece il tessuto connettivo reale della rete sociale Vinted.
La funzione analizza ogni titolo, normalizza la stringa e verifica la presenza delle keyword. In caso di assenza di corrispondenze, incrementa la categoria

residuale *Abbigliamento/Generico*. Al termine, i conteggi vengono ripuliti rimuovendo le categorie generiche, dopodiché la scelta del tag procede secondo una logica gerarchica: generalista puro, generalista sottosoglia, specialista puro o multi-categoria. Se le occorrenze di parole-chiave risultano poche o sporadiche, l'utente viene classificato come *Generalista* o come *Debole_X*, evitando sovra-interpretazioni. Quando due categorie risultano entrambe rilevanti – con la seconda che supera il 40% della prima – l'utente viene etichettato come *Multi_category*. Questo permette di preservare informazione cruciale per lo studio del ruolo che tali nodi svolgono nell'interconnessione delle comunità. Il Listing 2.3 mostra un estratto esteso della funzione, comprensivo delle soglie e della logica di decisione.

```
def assign_community_tag(item_titles_list: List[str]) -> Tuple[
    ↪ str, str]:
    # Parametri euristici di classificazione
    MIN_ABS_FOR_DOMINANT = 3
    MULTICATEGORY_RATIO = 0.4
    MULTICATEGORY_MIN_ABS = 2
    MIN_TOTAL_MATCHES = 4

    if not item_titles_list:
        return "Inattivo", "Inattivo"

    # Analisi dei conteggi keyword (codice omesso per brevit )
    # ...

    # Filtraggio delle categorie rilevanti
    interesting_counts = {
        k: v for k, v in counts.items()
        if k not in ["Abbigliamento/Generico", "Generalista"]
    ↪ and v > 0
    }

    # 1. GENERALISTA PURO: Segnale tematico assente o
    ↪ insufficiente
    if not interesting_counts or total_items_tagged <
    ↪ MIN_TOTAL_MATCHES:
        return "Generalista", "Generalista_Puro"
```

```
23
24      # Ordinamento per frequenza decrescente
25      sorted_counts = sorted(interesting_counts.items(), key=
    ↪ lambda x: x[1], reverse=True)
26      dominant_tag, dominant_count = sorted_counts[0]
27      second_tag, second_count = (sorted_counts[1] if len(
    ↪ sorted_counts) > 1 else ("Nessuno", 0))
28
29      # 2. GENERALISTA SOTTOSOGLIA (Segnale debole)
30      if dominant_count < MIN_ABS_FOR_DOMINANT:
31          return "Generalista", f"Debole_{dominant_tag}"
32
33      # 3. NICCHIA VERA O MULTI-CATEGORIA
34      # Se la seconda categoria    rilevante (> 40% della prima)
    ↪ e assoluta >= 2
35      if second_count >= MULTICATEGORY_MIN_ABS and \
36         (second_count / dominant_count) >= MULTICATEGORY_RATIO:
37          return "Multi_category", f"Multi_category_{dominant_tag
    ↪ }_{second_tag}"
38      else:
39          # Specialista verticale (es. Puro_Carte_TCG)
40          return dominant_tag, f"Puro_{dominant_tag}"
```

Listing 2.3: Estratto della funzione di tagging semantico con logica Multi-Category

Questo meccanismo rende il tagging riproducibile e comparabile tra utenti, eliminando ambiguità e automatizzando un'operazione che sarebbe altrimenti manuale e soggettiva.

2.4.2 Logica di Crawling, Robustezza e Self-Healing

L'estrazione dei dati avviene tramite la funzione `scrape_data`, che incorpora tre fasi distinte: (1) caricamento dello stato precedente, (2) scraping completo dei Seed e acquisizione degli archi della rete, (3) tagging semplificato di tutti i nodi incontrati. La presenza di tre fasi consente sia continuità dell'esecuzione in caso di interruzioni, sia una separazione netta tra l'espansione del grafo e l'analisi semantica dei nodi.

Fase 1 — Resume e caricamento dello stato. Prima di iniziare lo scraping, la funzione verifica l'esistenza di file precedentemente salvati: il dataset delle transazioni e il file dei tag. Se presenti, vengono caricati e integrati nella sessione corrente. Questo meccanismo non solo riduce il rischio di duplicazione dei dati, ma permette di riprendere il lavoro esattamente dal punto in cui era stato interrotto. La funzione legge inoltre eventuali tag preesistenti per evitare di analizzare utenti già processati.

Fase 2 — Acquisizione degli archi tramite i Seed. La seconda fase costituisce la parte più intensiva della pipeline. Per ciascun Seed, il sistema itera sulle pagine dei feedback tramite l'endpoint `/user_feedbacks`, estraendo gli ID dell'acquirente, del venditore e dell'oggetto coinvolto. Ogni transazione osservata viene salvata nel dataset, e ogni nuovo utente incontrato viene aggiunto al set dei nodi da processare in seguito. In parallelo, i titoli degli articoli associati ai feedback vengono accumulati in una lista dedicata al tagging del Seed.

Per ridurre la probabilità di blocchi o rate limiting, ogni ciclo utilizza pause casuali tra una pagina e l'altra, simulando un comportamento realistico. Se l'API restituisce un errore 403, indicativo di un blocco temporaneo, l'esecuzione non procede oltre: i dati vengono salvati e lo script termina in modo controllato. Al termine dell'elaborazione di ogni Seed, la funzione esegue un salvataggio incrementale. Un estratto della logica è riportato nel Listing 2.4.

```
# ============================================================
# FASE 2: ACQUISIZIONE ARCHI (Solo dai SEED)
# ============================================================
for current_user_id in seed_users_list:

    # Meccanismo di Idempotenza con Recovery
    if current_user_id in user_main_tags:
        # Se l'utente era stato marcato 'Inattivo' per errore, lo riprocessiamo
        if user_main_tags[current_user_id] == "Inattivo":
            print(f"Recovery: Utente {current_user_id} Inattivo -> Retagging.")
        else:
```

```
            continue # Skip se gia' processato correttamente

    # Skip se utente gia' taggato (Idempotenza)
    if current_user_id in user_main_tags:
        continue

    item_titles_for_tagging = []
    page = 1

    try:
        while True:
            # Chiamata API per ottenere feedback (transazioni)
            url_feedback = f"{vinted.api_url}/user_feedbacks?
↪ user_id={current_user_id}&page={page}..."
            response = vinted.scraper.get(url_feedback, ...)
            raw = response.json()

            # Estrazione transazioni
            for fb in raw.get("user_feedbacks", []):
                # ... (Logica di estrazione ID Acquirente/
↪ Venditore) ...
                item_titles_for_tagging.append(fb.get("
↪ item_title") or "")

            # Gestione paginazione
            if page >= raw.get("pagination", {}).get("
↪ total_pages", page):
                break

            page += 1
            # Pause casuali per Rate Limiting
            time.sleep(random.randint(SEED_MIN_SLEEP,
↪ SEED_MAX_SLEEP))

        # Calcolo tag al termine dello scarico
        main, detail = assign_community_tag(
↪ item_titles_for_tagging)
        user_main_tags[current_user_id] = main

    except Exception as e:
```

```
        if "403" in str(e):
            print("!!! BLOCCO 403 RILEVATO. SALVATAGGIO... !!!")
            save_progress(...)
            break

    # Salvataggio incrementale per sicurezza
    save_progress(...)
```

Listing 2.4: Crawling Seed: iterazione paginata e gestione errori

```
Analizzo SEED ID: 276534491
2025-11-04 11:14:27 - urllib3.connectionpool - DEBUG - https://www.vinted.it:443 "GET /api/v2/user_feedbacks?user_id
=276534491&page=1&per_page=20&by=all HTTP/11" 200 None
 -> Pausa di 17s tra le pagine di feedback...
2025-11-04 11:14:44 - urllib3.connectionpool - DEBUG - https://www.vinted.it:443 "GET /api/v2/user_feedbacks?user_id
=276534491&page=2&per_page=20&by=all HTTP/11" 200 None
 -> Pausa di 14s tra le pagine di feedback...
2025-11-04 11:14:58 - urllib3.connectionpool - DEBUG - https://www.vinted.it:443 "GET /api/v2/user_feedbacks?user_id
=276534491&page=3&per_page=20&by=all HTTP/11" 200 None
 -> Pausa di 11s tra le pagine di feedback...
2025-11-04 11:15:10 - urllib3.connectionpool - DEBUG - https://www.vinted.it:443 "GET /api/v2/user_feedbacks?user_id
=276534491&page=4&per_page=20&by=all HTTP/11" 200 None
 -> Pausa di 14s tra le pagine di feedback...
2025-11-04 11:15:24 - urllib3.connectionpool - DEBUG - https://www.vinted.it:443 "GET /api/v2/user_feedbacks?user_id
=276534491&page=5&per_page=20&by=all HTTP/11" 200 None
 -> Pausa di 19s tra le pagine di feedback...
2025-11-04 11:15:43 - urllib3.connectionpool - DEBUG - https://www.vinted.it:443 "GET /api/v2/user_feedbacks?user_id
=276534491&page=6&per_page=20&by=all HTTP/11" 200 None
 -> Pausa di 18s tra le pagine di feedback...
2025-11-04 11:16:01 - urllib3.connectionpool - DEBUG - https://www.vinted.it:443 "GET /api/v2/user_feedbacks?user_id
=276534491&page=7&per_page=20&by=all HTTP/11" 200 None
 -> Tag Assegnati al SEED: Main='Moda_Lusso', Detailed='Puro_Moda_Lusso'

--- SALVATAGGIO INCREMENTALE IN CORSO (23810 transazioni) ---
✅ Dati transazioni salvati in: vinted_raw_transactions.csv (23660 righe)
✅ Dati tag salvati in: vinted_user_tags.csv (3616 utenti)
PAUSA di 17s tra i SEED...
```

Figura 2.4: Log del completamento del tagging semantico dei Seed durante lo scraping.

Fase 3 — Tagging esteso dei nodi emersi. Una volta completata l'acquisizione degli archi, la rete contiene tutti gli utenti entrati in transazione con i Seed. La terza fase ha l'obiettivo di attribuire un tag semantico anche a questi nodi non-seed, garantendo una rappresentazione uniforme del grafo. Poiché lo scopo non è espandere ulteriormente la rete, ma solo classificare i profili, la funzione limita lo scraping a un massimo di tre pagine di feedback per utente, accelerando notevolmente il processo.

La fase integra un sistema di resume avanzato: se esistono file di tag precedenti, gli utenti già processati vengono saltati; inoltre, ogni dieci nodi viene effettuato un salvataggio incrementale. In caso di errore 401, sintomo tipico di cookie

scaduto o sessione invalidata, lo script effettua un riavvio controllato una sola volta; il riavvio controllato è stato introdotto successivamente per due ragioni: la prima è che la comparsa di questo errore propaga in cascata il tag Inattivo alle successive chiamate HTTP marcando un nodo come Inattivo che in realtà poteva non esserlo; la seconda è più ovvia e permette il riavvio dello script quando si è impossibilitati nel farlo manualmente. Ulteriori errori portano al tagging conservativo del nodo come *Inattivo*. Il Listing 2.5 riassume la logica essenziale.

```
# ============================================================
# FASE 3: ACQUISIZIONE TAGGING
# ============================================================
...
        # FASE DI RECUPERO: se esistono utenti Inattivi,
    ↪ ritaggiamo solo quelli

        if retry_inactive_only:
            try:
                df_tags = pd.read_csv(TAGS_FILE_NAME)
                if 'Node_ID' in df_tags.columns and 'Main_Tag'
    ↪ in df_tags.columns:
                    inactive_users = (
                        df_tags[df_tags['Main_Tag'].astype(str)
    ↪ .str.strip().str.lower() == "inattivo"]
                        ['Node_ID']
                        .dropna()
                        .astype(int)
                        .tolist()
                    )
                    if inactive_users:
                        print(f" Modalita' RECUPERO ATTIVA
    ↪ {len(inactive_users)} utenti 'Inattivo' verranno
    ↪ ritaggati.")
                        nodes_to_tag = inactive_users  #
    ↪ sovrascrive la lista da processare
                    else:
                        print(" Nessun utente 'Inattivo'
    ↪ trovato. Procedo normalmente.")
                else:
```

```
                print(" File tag privo di colonne attese ('Node_ID', 'Main_Tag'). Procedo normalmente.")
            except Exception as e:
                print(f"Errore durante il caricamento utenti inattivi: {e}")

        # --- CICLO DI TAGGING ---
        save_counter_fase3 = 0

        for i, user_id in enumerate(nodes_to_tag):
            item_titles_for_tagging: List[str] = []
            try:
                page = 1
                while True:
                    url_feedback = f"{vinted.api_url}/user_feedbacks?user_id={user_id}&page={page}&per_page=20&by=all"
                    response = vinted.scraper.get(url_feedback,
                                                  headers=getattr(vinted, "headers", {}),
                                                  cookies=getattr(vinted, "cookies", {}),
                                                  proxies=getattr(vinted, "proxy", None))
                    response.raise_for_status()
                    raw_json_data = response.json()

                    if 'user_feedbacks' not in raw_json_data or not raw_json_data['user_feedbacks']:
                        break  # Nessun feedback trovato

                    for feedback in raw_json_data['user_feedbacks']:
                        item_title = feedback.get('item_title') or ""
                        item_titles_for_tagging.append(item_title)

                    pagination = raw_json_data.get('pagination')
```

```
                    if not pagination or pagination.get('
    ↪ current_page', page) >= pagination.get('total_pages',
    ↪ page) or page >= 3:
                        break
                    page += 1
                    time.sleep(random.randint(5, 10))  # Pausa
    ↪ breve tra le pagine

                main_tag, detailed_tag = assign_community_tag(
    ↪ item_titles_for_tagging)
                user_main_tags[user_id] = main_tag
                user_detailed_tags[user_id] = detailed_tag
                print(f" -> Tag Assegnati: Main='{main_tag}',
    ↪ Detailed='{detailed_tag}'")

            except Exception as e:
                err_str = str(e)
                print(f"Errore durante il tagging di {user_id}:
    ↪  {err_str}")
                if ("401" in err_str or "Unauthorized" in
    ↪ err_str) and not HAS_RESTARTED:
                    ...
```

Listing 2.5: Tagging estensivo dei nodi scoperti (Fase 3)

```
Tagging Utente 49/26356 (ID: 50594082)
2025-11-04 12:30:32 - urllib3.connectionpool - DEBUG - https://www.vinted.it:443 "GET /api/v2/user_feedbacks?user_id
=50594082&page=1&per_page=20&by=all HTTP/11" 200 None
2025-11-04 12:30:39 - urllib3.connectionpool - DEBUG - https://www.vinted.it:443 "GET /api/v2/user_feedbacks?user_id
=50594082&page=2&per_page=20&by=all HTTP/11" 200 None
2025-11-04 12:30:46 - urllib3.connectionpool - DEBUG - https://www.vinted.it:443 "GET /api/v2/user_feedbacks?user_id
=50594082&page=3&per_page=20&by=all HTTP/11" 200 None
 -> Tag Assegnati: Main='Action_Figure', Detailed='Puro_Action_Figure'
PAUSA di 18s tra il tagging...

Tagging Utente 50/26356 (ID: 240779564)
2025-11-04 12:31:04 - urllib3.connectionpool - DEBUG - https://www.vinted.it:443 "GET /api/v2/user_feedbacks?user_id
=240779564&page=1&per_page=20&by=all HTTP/11" 200 None
 -> Tag Assegnati: Main='Generalista', Detailed='Generalista'

--- SALVATAGGIO INCREMENTALE IN CORSO (27857 transazioni) ---
✅ Dati transazioni salvati in: vinted_raw_transactions.csv (27497 righe)
✅ Dati tag salvati in: vinted_user_tags.csv (3643 utenti)
PAUSA di 17s tra il tagging...

Tagging Utente 51/26356 (ID: 43385136)
2025-11-04 12:31:21 - urllib3.connectionpool - DEBUG - https://www.vinted.it:443 "GET /api/v2/user_feedbacks?user_id
=43385136&page=1&per_page=20&by=all HTTP/11" 200 None
2025-11-04 12:31:28 - urllib3.connectionpool - DEBUG - https://www.vinted.it:443 "GET /api/v2/user_feedbacks?user_id
=43385136&page=2&per_page=20&by=all HTTP/11" 200 None
2025-11-04 12:31:37 - urllib3.connectionpool - DEBUG - https://www.vinted.it:443 "GET /api/v2/user_feedbacks?user_id
=43385136&page=3&per_page=20&by=all HTTP/11" 200 None
 -> Tag Assegnati: Main='Action_Figure', Detailed='Puro_Action_Figure'
PAUSA di 13s tra il tagging...
```

Figura 2.5: Log della fase di tagging leggero dei nodi non-Seed (prima parte).

Figura 2.6: Log della fase di tagging leggero dei nodi non-Seed (seconda parte).

2.4.3 Output della Prima Fase

L'intera procedura descritta conduce alla generazione di due file fondamentali per la costruzione del dataset finale:

- `vinted_raw_transactions.csv`, contenente tutte le transazioni osservate nel crawling iniziale, rappresentate come archi direzionati $A \rightarrow V$;
- `vinted_user_tags.csv`, che mappa ciascun ID utente ai relativi attributi semantici (*Main Tag* e *Detailed Tag*).

Questi due artefatti rappresentano il nucleo del dataset ASNM. Nelle prossime sezioni verranno integrati, puliti e trasformati in un'unica struttura coerente, attraverso un processo di normalizzazione e ricostruzione del grafo. Sarà inoltre mostrato come ulteriori script – `dataset_construction.py` e `get_recensioni_venditori_FINAL.py` – completino il ciclo di produzione portando alla generazione del dataset finale utilizzato nelle analisi e negli esperimenti empirici.

2.5 Pulizia e Costruzione del Dataset Intermedio

I dati grezzi ottenuti dalla precedente fase di acquisizione, pur essendo ricchi dal punto di vista informativo, non possono essere utilizzati direttamente per un'analisi di rete affidabile. Presentano infatti nodi privi di significato semantico, account inattivi o borderline, transazioni con utenti eliminati e, soprattutto, mancano della componente fondamentale per la costruzione della matrice di fiducia: la reciprocità del feedback.

In questa sezione viene descritta in modo organico la pipeline di post–processing implementata per trasformare tali dati nel *dataset finale*. La procedura si articola in due fasi consecutive: una prima fase di pulizia e normalizzazione (`dataset_construction.py`) e una seconda fase di arricchimento attraverso il recupero delle contro–recensioni (`get_recensioni_venditori_FINAL.py`).

La prima parte della pipeline è dedicata all'eliminazione del rumore e alla preparazione di una struttura coerente su cui operare in seguito. Lo script `dataset_construction.py` carica i dati estratti in precedenza, normalizza gli identificativi e applica la logica di filtraggio dei nodi basata sui tag semantici.

2.5.1 Filtraggio dei Nodi

Durante la fase di tagging descritta nel capitolo precedente è emersa un'ampia porzione di utenti caratterizzati da attività minima o da una semantica non informativa. Per evitare che tali nodi distorcano le analisi successive e aumentino artificialmente la densità della rete, è stata adottata una strategia di filtraggio.

In questa strategia, gli utenti etichettati come `Inattivo`, `Generalista_Puro` con un'unica eccezione: i nodi *Seed*, che rappresentano i punti di ancoraggio della rete e devono essere preservati, essi risultano immuni.

```
# Filtraggio dei nodi con Strategia "Deadline"
filtro_seed = df_tags["Node_ID"].isin(SEED_USER_IDS)

filtro_inattivo = (df_tags["Main_Tag"] == "Inattivo")
filtro_puro = (df_tags["Detailed_Tag"] == "Generalista_Puro")
```

```
filtro_vecchio_ambiguo = (
    (df_tags["Main_Tag"] == "Generalista") &
    (df_tags["Detailed_Tag"] == "Generalista")
)

filtro_drop_totale = (filtro_inattivo | filtro_puro |
    ↪ filtro_vecchio_ambiguo)
nodi_da_droppare = df_tags[filtro_drop_totale & (~filtro_seed)]
nodi_da_tenere_df = df_tags.drop(nodi_da_droppare.index)
```

Listing 2.6: Logica di filtraggio dei nodi in dataset_construction.py

Terminato il filtraggio, viene generato il set dei nodi ammessi, che costituisce la base del grafo.

2.5.2 Merge e Generazione del Dataset Intermedio

Una volta definito il set di nodi validi, anche la struttura delle transazioni viene purificata. Sono mantenuti esclusivamente gli archi in cui entrambi gli utenti coinvolti appartengono al set dei nodi ammessi; ciò elimina transazioni spurie con utenti non più presenti o non significativi.
Infine, vengono uniti i tag semantici alle due estremità di ogni arco, producendo così il file intermedio `vinted_dataset_PULITO.csv`. Questo dataset contiene già tutte le informazioni strutturali necessarie, ma conserva ancora una limitazione: il rating disponibile è solamente quello dell'acquirente verso il venditore ($A \to V$).

2.6 Recupero delle Contro–Recensioni

La seconda fase della pipeline ha l'obiettivo di completare la reciprocità della fiducia recuperando la contro–recensione del venditore verso l'acquirente ($V \to A$). Tale informazione non è disponibile tramite la route API utilizzata nella fase di scraping principale, e richiede quindi una procedura dedicata.

2.6.1 Motivazione della Suddivisione in Due Script

È importante sottolineare che, inizialmente, si era tentato di integrare il recupero della contro–recensione direttamente nella fase di tagging. Tuttavia, questa configurazione si è rivelata estremamente instabile: alternare chiamate API sul venditore e sull'acquirente all'interno dello stesso flusso aumentava in modo drastico il numero di errori di rete, introduceva ritardi significativi e rallentava l'intero processo.

Per migliorare robustezza ed efficienza è stata quindi adottata una divisione metodologica in due blocchi operativi distinti. La prima fase si occupa esclusivamente del tagging semantico, risultando più veloce e stabile; la seconda è invece dedicata al recupero delle contro–recensioni tramite chiamate mirate, effettuate solo quando il dataset è stato ormai ripulito. Questa intuizione ha ridotto i tempi totali di esecuzione e ha aumentato la resilienza dell'intera pipeline.

2.6.2 Ricerca Mirata della Contro–Recensione

Lo script `get_recensioni_venditori_FINAL.py` itera su ogni riga del dataset intermedio e, per ciascuna coppia (A, V), interroga direttamente il profilo dell'acquirente, scorrendo i feedback ricevuti fino a trovare (eventualmente) il rating lasciato dal venditore. La funzione centrale di questa fase è riportata nel Listing seguente.

```
1  def find_counter_review(vinted_client: Vinted, buyer_id: int,
       ↪ seller_id: int) -> int | None:
2      page = 1
3      while True:
4          url_feedback = f"{vinted_client.api_url}/user_feedbacks
       ↪ ?user_id={buyer_id}&page={page}&per_page=20&by=all"
5          response = vinted_client.scraper.get(url_feedback, ...)
6          response.raise_for_status()
7
8          raw_json_data = response.json()
9          if 'user_feedbacks' not in raw_json_data or not
       ↪ raw_json_data['user_feedbacks']:
10             return None
11
```

```
        for feedback in raw_json_data['user_feedbacks']:
            if feedback.get('feedback_user_id') == seller_id:
                return feedback.get('rating')

        pagination = raw_json_data.get('pagination')
        if not pagination or pagination["current_page"] >=
    ↪ pagination["total_pages"]:
            return None

        page += 1
        time.sleep(random.randint(2, 5))
```

Listing 2.7: Funzione di recupero della contro-recensione

Poiché ogni transazione richiede una query dedicata, questa fase introduce un elevato numero di chiamate API. Per preservare l'esecuzione anche in presenza di errori transitori — disconnessioni VPN, errori 401/500/503 o timeout — è stato reintrodotto, per ovvie ragioni, il meccanismo di *self–healing*: lo script salva lo stato corrente e si riavvia automaticamente quando possibile.

2.7 Dataset Finale

La pipeline produce infine il file `vinted_dataset_FINAL.csv`, che rappresenta la versione definitiva del dataset da cui verrà costruito il grafo e calcolate le metriche di rete del Capitolo 4. Ogni riga del file descrive un'interazione completa tra due utenti e contiene:

- gli identificativi dell'acquirente e del venditore;
- la recensione fornita dall'acquirente ($A \to V$);
- la contro–recensione eventualmente recuperata ($V \to A$), oppure un valore nullo in assenza di riscontro;
- i tag semantici principali e dettagliati per entrambi i nodi.

L'insieme di queste informazioni permette di costruire una rete direzionale, pesata e semanticamente arricchita, adatta a un'analisi strutturale approfondita della fiducia tra utenti.

Capitolo 3

Modellazione del Grafo e Analisi ASNM

Dopo aver completato la fase di acquisizione e pulizia dei dati, il passo successivo consiste nella trasformazione del dataset tabellare in una struttura di rete analizzabile. In questo capitolo viene descritto in dettaglio il processo di costruzione del grafo orientato e pesato, l'implementazione delle metriche di analisi topologica e semantica e l'interpretazione dei risultati ottenuti.
Tutte le procedure descritte sono state implementate nello script `asnm_analysis.py`, utilizzando la libreria `NetworkX` per la manipolazione del grafo e `python-louvain` per la community detection.

3.1 Costruzione del Grafo Orientato e Pesato

Il grafo è stato modellato come un **digrafo pesato** $G = (V, E)$, pensato per catturare la natura asimmetrica delle transazioni e la reciprocità del feedback.

3.1.1 Nodi e Archi

Nodi (V): Rappresentano utenti univoci presenti nel dataset finale. Ad ogni nodo sono stati associati attributi semantici derivati dalla fase di tagging (`main_tag`, `detailed_tag`), per consentire analisi correlate al contenuto.

Archi (E): Le relazioni sono rappresentate come archi direzionati:

- $A \to V$: feedback lasciato dall'acquirente verso il venditore.

- $V \to A$: contro-recensione del venditore verso l'acquirente (se disponibile).

Modellazione Asimmetrica dei Pesi (W): La topologia del grafo è stata costruita adottando una strategia di pesatura differenziata per disaccoppiare la componente economica da quella reputazionale. Questa scelta distingue tra due forme di interazione:

- **Fiducia Implicita (Arco $A \to V$):** Il peso w corrisponde al **Volume di Transazioni** (numero di oggetti acquistati, tipicamente 1). Questo arco rappresenta l'*atto economico*: l'acquirente "investe" sul venditore, ma il segnale è puramente quantitativo.
- **Fiducia Esplicita (Arco $V \to A$):** Il peso w corrisponde al **Rating** (scala 1-5). Questo arco rappresenta l'*atto sociale*: il venditore conferma la validità della controparte. In un contesto C2C, il feedback di ritorno è un segnale di "validazione dell'identità" molto più forte della semplice transazione. È tuttavia necessario osservare che questo prezioso segnale non è sempre disponibile: la tendenza di molti venditori a non ricambiare il feedback rappresenta un limite informativo che sottostima la reale densità della rete di fiducia.

Questa struttura offre un vantaggio analitico cruciale per la robustezza del grafo contro pattern anomali (es. *Scamming* o *Wash Trading*). Un nodo che genera elevati volumi di acquisto (alto Out-Degree pesato in andata) ma non riceve feedback di ritorno di qualità (assenza di archi pesati in ritorno) viene naturalmente penalizzato dagli algoritmi di centralità, impedendo che un'attività puramente volumetrica venga confusa con una reale reputazione sociale.

3.1.2 Implementazione in NetworkX

La funzione `build_graph` scansiona il dataset e costruisce il grafo. È stata gestita la presenza condizionale delle contro-recensioni: mentre il feedback dell'acquirente è sempre presente, quello del venditore viene aggiunto solo se disponibile.

```python
def build_graph(df: pd.DataFrame) -> nx.DiGraph:
    G = nx.DiGraph()

    for _, row in df.iterrows():
        buyer_id = int(row["Acquirente_ID"])
        seller_id = int(row["Venditore_ID"])

        # Aggiunta Nodi con Metadati Semantici
        G.add_node(buyer_id, main_tag=row["Main_Tag_Acquirente"
    ↪ ])
        G.add_node(seller_id, main_tag=row["Main_Tag_Venditore"
    ↪ ])

        # Arco A -> V (Feedback Acquirente)
        G.add_edge(
            buyer_id, seller_id,
            weight=row.get("Numero_Transazioni", 1),
            rating=row.get("Rating_Acquirente_V", 0)
        )

        # Arco V -> A (Contro-recensione, se esiste)
        rating_v_a = row.get("Rating_Venditore_A", 0)
        if pd.notna(rating_v_a) and rating_v_a > 0:
            G.add_edge(
                seller_id, buyer_id,
                weight=rating_v_a
            )

    return G
```

Listing 3.1: Costruzione del Digrafo in NetworkX

3.2 Metriche Strutturali e Analisi di Centralità

Per caratterizzare la topologia della rete Vinted, sono state calcolate metriche macroscopiche fondamentali che descrivono la coesione globale e la natura delle

interazioni. L'analisi è stata condotta su un grafo finale composto da **16.533 nodi** e **25.215 archi**.
Dai risultati computazionali emergono evidenze strutturali chiave:

- **Reciprocità Elevata (≈ 0.67):** Il calcolo ha restituito un indice di reciprocità pari a 0.67. Questo valore è straordinariamente alto per una rete commerciale e suggerisce che, su Vinted, la norma sociale prevede lo scambio mutuo di feedback. A differenza di piattaforme B2C unidirezionali, qui la relazione tende a stabilizzarsi come bidirezionale ($A \leftrightarrow B$), confermando l'ipotesi di un "Social Commerce" basato sulla fiducia reciproca.

- **Clustering Coefficient Quasi Nullo (0.00006):** Nonostante l'alta reciprocità, il coefficiente di clustering medio è infinitesimale. Questo dato, apparentemente contraddittorio, descrive in realtà perfettamente la topologia **Hub-and-Spoke**: gli acquirenti ("Spokes") sono connessi ai venditori ("Hubs"), ma non interagiscono tra loro. Mancano quasi totalmente le strutture a "triangolo" tipiche delle reti di amicizia, confermando che la coesione sociale è verticale (tra pari commerciali) e non orizzontale (tra clienti).

- **Densità e Jaccard Similarity (0.0):** La rete si conferma estremamente sparsa, con una similarità di Jaccard media nulla. Ciò indica che non vi è sovrapposizione significativa tra i vicini dei nodi connessi: ogni venditore tende ad avere la propria "bolla" di acquirenti esclusivi, con poche intersezioni strutturali.

Tutto ciò va a confermare che Vinted è una rete commerciale e sì sociale, ma di mercato, non di amicizia.
Per identificare gli attori chiave in questo ecosistema frammentato, sono state calcolate le metriche di centralità puntuali:

- **Degree Centrality (In/Out):** Utilizzata per distinguere i "Power Seller" (alto In-Degree) dagli acquirenti compulsivi.

- **PageRank:** Calcolato con smorzamento $\alpha = 0.85$, per premiare non solo chi vende tanto, ma chi riceve feedback da utenti a loro volta attivi. In sostanza il Pagerank premia la qalità del feedback ricevuto.

- **Betweenness Centrality:** Data la dimensione del grafo, è stata utilizzata un'approssimazione con $k = 1000$ pivot. Questa metrica si è rivelata cruciale per individuare i nodi "ponte" che, pur non essendo necessariamente i più grandi venditori, connettono cluster altrimenti isolati.

```
def calculate_centralities(G: nx.DiGraph, k: int, seed: int):
    in_degree = nx.in_degree_centrality(G)
    out_degree = nx.out_degree_centrality(G)
    pagerank = nx.pagerank(G, alpha=0.85)
    betweenness = nx.betweenness_centrality(G, k=k, seed=seed,
    ↪ normalized=True)
```

Listing 3.2: Calcolo delle centralità con approssimazione della Betweenness

3.3 Rilevazione delle Comunità e Modularità

Uno degli obiettivi primari del progetto era verificare se la rete Vinted, al di là delle singole transazioni, si auto-organizzasse in strutture latenti basate sugli interessi. Per testare questa ipotesi, è stato applicato l'algoritmo di **Louvain**, ottimizzato per massimizzare la modularità su reti di grandi dimensioni.
L'algoritmo ha identificato una partizione della rete in **66 comunità distinte**, raggiungendo un valore di Modularità (Q) pari a **0.9489**. Questo risultato è eccezionalmente significativo: in Social Network Analysis, valori di $Q > 0.7$ indicano una struttura di comunità molto forte. Un valore di 0.95 suggerisce che la rete è quasi perfettamente segregata in "isole" tematiche, con densissime connessioni interne e pochissimi collegamenti verso l'esterno.

Nota Metodologica: Sebbene il grafo sia nativamente modellato come diretto (*Directed*), per l'applicazione di Louvain esso è stato trattato come non orientato (`G.to_undirected()`). Questa operazione è necessaria poiché la definizione standard di modularità si basa sulla densità dei legami, interpretan-

do ogni transazione (indipendentemente dalla direzione) come un segnale di affinità tematica tra i nodi coinvolti.

La successiva fase di *Naming Semantico* ha confermato la natura di questi cluster: le comunità rilevate non sono artefatti matematici, ma corrispondono a nicchie di mercato reali (es. *Moda*, *TCG*, *Action Figures*), isolate tra loro ma tenute insieme da una rete sottile di broker multi-categoria.

```
def run_community_detection(G: nx.DiGraph, seed: int):
    # Conversione a grafo non orientato per Louvain
    G_undir = G.to_undirected()

    # Partizionamento ottimo
    partition = community_louvain.best_partition(G_undir,
    ↪ random_state=seed)
    modularity = community_louvain.modularity(partition,
    ↪ G_undir)

    return partition, modularity
```

Listing 3.3: Esecuzione dell'algoritmo di Louvain

3.4 Naming Semantico e Profilazione delle Comunità

L'output grezzo dell'algoritmo di Louvain è una partizione matematica in cui ogni nodo è associato a un ID numerico di comunità (es. *Community 0, Community 1*). Sebbene ottimali dal punto di vista della modularità, questi identificativi sono privi di significato semantico e non permettono di comprendere la natura commerciale del gruppo.

Per tradurre la struttura topologica in informazioni di business interpretabili, è stata sviluppata una procedura di **Naming Automatico** (*Smart Labeling*). La funzione `naming_communities` analizza la composizione demografica di ciascun cluster e assegna un'etichetta descrittiva basata su tre livelli di logica decisionale.

3.4.1 Logica di Assegnazione dell'Etichetta

L'algoritmo non si limita ad assegnare il tag più frequente (approccio *Naive*), che rischierebbe di etichettare quasi tutte le comunità come "Generalista" data la prevalenza numerica di tale categoria. Al contrario, adotta un approccio gerarchico per estrarre il "segnale" tematico dal "rumore":

1. **Estrazione del Segnale (Noise Filtering):** L'algoritmo calcola le frequenze dei *Main Tag* all'interno della comunità ignorando deliberatamente le etichette generiche (`Generalista`). Questo permette di identificare la "nicchia dominante" (es. *Fumetti_Manga*) anche se questa rappresenta una minoranza rispetto alla massa di utenti non specializzati.

2. **Gruppi Ibridi (Logica MIX):** Se la categoria dominante risulta essere `Multi_category`, l'algoritmo esegue un'analisi di secondo livello sui *Detailed Tags*. Invece di etichettare la comunità con un generico "Multi-Category", il sistema analizza quale specifica combinazione di categorie è prevalente (es. *Multi_category_Moda_Lusso_Tech_Elettronica*) e genera un nome composto dinamico (es. **"MIX: MODA & TECH"**). Questo passaggio è fondamentale per identificare le aree di interconnessione tra mercati diversi.

3. **Valutazione della Purezza (Suffisso Misto):** Viene calcolata la **Dominanza Percentuale** del tema principale. Se il tema dominante copre meno del 30% dei nodi della comunità, viene aggiunto il suffisso **"(Misto)"** all'etichetta (es. *"03 - MODA & LUSSO (Misto)"*). Questo garantisce rigore scientifico, distinguendo le *Echo Chambers* pure (dove tutti vendono la stessa cosa) dai cluster eterogenei tenuti insieme da legami deboli.

Nel Listing 3.4 è riportata l'implementazione della logica di naming, che normalizza anche le stringhe per la visualizzazione finale su Gephi.

```
1 def naming_communities(df_nodes: pd.DataFrame) -> pd.DataFrame:
2     community_names = {}
3     grouped = df_nodes.groupby('Community')
4
```

```
    for comm_id, group in grouped:
        counts = group['main_tag'].value_counts()
        total_nodes = len(group)

        if not counts.empty:
            # 1. ESTREZIONE DEL SEGNALE
            # Cerchiamo il primo tag che NON sia rumore (Generalista/Inattivo)
            top_tag = "Generalista"
            for tag in counts.index:
                if tag not in ["Generalista", "Sconosciuto", "Inattivo"]:
                    top_tag = tag
                    break

            # Calcolo della Dominanza (Purezza del cluster)
            dominance_perc = (counts[top_tag] / total_nodes) * 100

            # Pulizia stringa base
            clean_name = top_tag.replace("Puro_", "").replace("Debole_", "").replace("Generalista_", "")

            # 2. DRILL-DOWN SUI GRUPPI IBRIDI (MIX)
            if "Multi_category" in clean_name:
                # Analisi dei Detailed Tags per capire CHE TIPO di mix
                detailed_counts = group[group['main_tag'] == top_tag]['detailed_tag'].value_counts()

                if not detailed_counts.empty:
                    top_detailed = detailed_counts.index[0]
                    # Trasformazione: "Multi_category_Moda_Tech " -> "MIX: MODA & TECH"
                    short_mix = top_detailed.replace("Multi_category_", "").replace("_", " & ")
                    clean_name = f"MIX: {short_mix}"
                else:
                    clean_name = "MULTI-CATEGORY MIX"

```

```
37            # Mapping categorie standard (es. Moda -> MODA &
    ↪ LUSSO)
38            elif "Moda" in clean_name: clean_name = "MODA &
    ↪ LUSSO"
39            elif "Carte" in clean_name: clean_name = "TRADING
    ↪ CARD GAMES"
40            # ... (altri mapping omessi) ...
41            else: clean_name = clean_name.upper()
42
43            # 3. CONTROLLO PUREZZA
44            # Aggiunge "(Misto)" se la dominanza    < 30% e non
    ↪     gi   un gruppo MIX
45            suffix = ""
46            if dominance_perc < 30 and "MIX" not in clean_name:
47                suffix = " (Misto)"
48
49            # Label finale ordinata (es. "12 - TRADING CARD
    ↪ GAMES")
50            label = f"{str(comm_id).zfill(2)} - {clean_name}{
    ↪ suffix}"
51        else:
52            label = f"{comm_id} - Sconosciuto"
53
54        community_names[comm_id] = label
55
56    # Applicazione del mapping al DataFrame
57    df_nodes['Community_Label'] = df_nodes['Community'].map(
    ↪ community_names)
58    return df_nodes
```

Listing 3.4: Logica di Naming Semantico

Questa procedura permette di trasformare i risultati astratti dell'analisi topologica in una mappa semantica del mercato, rendendo immediatamente visibile la distinzione tra cluster tematici verticali e zone di confine.

3.5 Modelli di Diffusione e Massimizzazione dell'Influenza

Dopo aver caratterizzato la struttura statica della rete Vinted nelle sezioni precedenti, l'analisi si è focalizzata sulla dinamica dei processi di propagazione della fiducia e della reputazione attraverso le relazioni di feedback. L'obiettivo è identificare gli utenti più influenti, cioè coloro che massimizzano la diffusione di informazioni positive all'interno della rete.
A tale scopo è stato adottato il modello **Independent Cascade (IC)**, accompagnato dall'algoritmo di ottimizzazione greedy CELF per la selezione dei nodi seed.

3.5.1 Modello di Diffusione: Independent Cascade

Il modello **Independent Cascade** simula la propagazione stocastica della fiducia:

- Al tempo $t = 0$, un insieme di nodi iniziali (*Seeds*) viene attivato. Questi rappresentano utenti influenti o altamente affidabili.
- Ad ogni step $t + 1$, ciascun nodo attivato prova una singola volta a influenzare ciascun vicino inattivo con probabilità p.
- Se il tentativo ha successo, il vicino diventa attivo al passo successivo; altrimenti non potrà più essere attivato da quel nodo.
- Il processo termina quando non ci sono nuovi nodi attivati.

Nel contesto di Vinted, questa simulazione rappresenta la diffusione della fiducia: un acquirente soddisfatto può indurre altri utenti a fidarsi dello stesso venditore, replicando dinamiche di passaparola commerciale.

3.5.2 Implementazione Monte Carlo

Poiché il processo è stocastico, è stato necessario ripetere molte simulazioni per ottenere una stima affidabile dello spread medio:

```python
def ic_spread_mc(G: nx.DiGraph, seeds: list, prob: float, mc:
    ↪ int) -> float:
     spread_sum = 0
     for _ in range(mc):
         active = set(seeds)
         newly_active = set(seeds)
         while newly_active:
             # ... (Logica di propagazione probabilistica) ...
             for node in newly_active:
                 for neighbor in G.successors(node):
                     if neighbor not in active and random.random
    ↪ () <= prob:
                         next_new.add(neighbor)
             # ...
         spread_sum += len(active)
     return spread_sum / mc
```

Listing 3.5: Simulazione Monte Carlo del modello Independent Cascade

Nel progetto è stata utilizzata una probabilità uniforme di attivazione $p = 0.1$ e un numero di iterazioni Monte Carlo $MC = 50$ per garantire la stabilità statistica dei risultati.

3.6 Strategie di Selezione dei Seed

L'obiettivo della *Influence Maximization* è trovare un insieme S di $k = 5$ nodi che massimizza lo spread atteso $\sigma(S)$. Poiché il problema è NP-Hard, sono state adottate tre strategie di selezione per confronto:

1. **Top Out-Degree:** seleziona i nodi con il maggior numero di connessioni in uscita (Hubs strutturali).

2. **Top PageRank:** seleziona nodi con elevata centralità di autorità (Reputazione).

3. **CELF (Cost-Effective Lazy Forward):** algoritmo greedy ottimizzato, che riduce le valutazioni ridondanti sfruttando la proprietà submodulare della funzione di influenza.

3.6.1 Algoritmo CELF

L'algoritmo CELF riduce significativamente il costo computazionale rispetto al greedy standard, che richiede $O(k \cdot N \cdot R)$ valutazioni.

- Calcola lo spread iniziale per tutti i nodi candidati (solo al primo passo).
- Ordina i nodi in una coda di priorità decrescente.
- Ad ogni iterazione, ricalcola il guadagno marginale solo del nodo in testa (Lazy Evaluation).

```
def celf_algorithm(G: nx.DiGraph, k: int, prob: float, mc: int)
    ↪ :
     # ... (Calcolo spread iniziale e ordinamento) ...
     while len(seeds) < k:
         matched = False
         while not matched and gains:
             current_best_gain, current_best_node = gains[0]
             # Ricalcolo marginale solo per il top della coda
             new_spread = ic_spread_mc(G, seeds + [
    ↪ current_best_node], prob, mc)
             marginal_gain = new_spread - current_spread

             # Se il nodo rimane il migliore anche dopo il
    ↪ ricalcolo, lo scegliamo
             if not gains or marginal_gain >= gains[0][0]:
                 seeds.append(current_best_node)
                 matched = True
             else:
                 # Altrimenti lo riposizioniamo nella coda
                 gains.append((marginal_gain, current_best_node)
    ↪ )
                 gains.sort(reverse=True)
     return seeds, current_spread
```

Listing 3.6: Implementazione dell'algoritmo CELF

3.7 Analisi dei Risultati e Interpretazioni

Dopo l'esecuzione delle simulazioni, i risultati evidenziano la superiorità dell'approccio Greedy/CELF rispetto alle euristiche statiche.

Strategia	**Spread Medio (Nodi attivati)**	**Top Seed Identificati**
CELF	**446.90**	[55125380, 138224980, 53097946, 76860837, 86438122]
Top Out-Degree	442.86	[55125380, 138224980, 53097946, 76860837, 86438122]
Top PageRank	381.68	[55125380, 162622596, 138224980, 86438122, 53097946]

Tabella 3.1: Confronto delle strategie IM ($k = 5, p = 0.1$).

Analisi dei Seed Selezionati: È interessante notare una sovrapposizione parziale tra gli insiemi di nodi selezionati.

- Il nodo **55125380** emerge come "Super-Influencer" indiscusso, venendo selezionato come primo seed da tutte e tre le strategie. Il suo spread marginale iniziale è di **167.16 nodi**, un valore eccezionale che lo qualifica come hub centrale della componente gigante.
- CELF e Out-Degree convergono sullo stesso set di nodi, ma CELF garantisce teoricamente (per la proprietà di submodularità) che questa combinazione sia locale ottima, mentre per l'Out-Degree è una coincidenza dovuta alla struttura specifica della rete.
- La strategia **PageRank** performa significativamente peggio (Spread 381 vs 446). Questo dimostra che l'*autorità* (essere linkati da nodi importanti) non si traduce automaticamente in *capacità di diffusione virale*. Nodi come **162622596**, pur avendo alto PageRank, probabilmente risiedono in zone della rete già coperte o "chiuse", offrendo un guadagno marginale nullo in termini di diffusione.

Interpretazione della dinamica di diffusione: La rete Vinted mostra caratteristiche di **rete a stella estesa**:

- I nodi centrali (venditori) fungono da hub naturali.

- La propagazione della fiducia segue percorsi radiali dai venditori verso gli acquirenti.

- La strategia CELF ha saputo sfruttare questa topologia selezionando nodi che coprono "raggi" (Spokes) diversi della rete, massimizzando la copertura globale e minimizzando la sovrapposizione dei follower.

```
(ANSM_env) C:\Users\l1par\OneDrive\Desktop\vinted_proj_ANSM>C:/Users/l1par/OneDrive/Desktop/vinted_proj_ANSM/ANSM_env/Scripts/python.exe c:/Users/l1par/OneDrive/Desktop/vinted_proj_ANSM/ANSM_env/asnm_analysis.py
==================================================
ASNM FINAL ANALYSIS - VINTED DATASET
==================================================
Caricamento vinted_dataset_FINAL.csv...
Costruzione del grafo in corso...
Grafo costruito: 16533 nodi, 25215 archi.
Calcolo metriche strutturali...
Calcolo Clustering Coefficient...
Calcolo Jaccard Similarity (su archi esistenti)...
--- STATS ---
Reciprocità: 0.67864
Clustering: 0.00006
Jaccard Similarity Media: 0.00000
Calcolo Degree Centrality (In/Out)...
Calcolo PageRank...
Calcolo Betweenness Centrality (Approssimata con k=1000)...
Eseguo Community Detection (Louvain)...
Louvain: Trovate 66 community. Mod: 0.9489
Generazione automatica nomi delle Community (Auto-Naming)...
Nomi assegnati. Esempio:
           Community   Community_Label
Node_ID
108622456          3  03 - GENERALISTA
79007304           1  01 - GENERALISTA
70840015           2  02 - CARTE & TCG
50565039           4  04 - GENERALISTA
183835223          5  05 - GENERALISTA

✅ Report nodi completo (con nomi) salvato.
Avvio algoritmo CELF (Target: 5 seeds, Prob: 0.1, MC: 50)...
CELF: Calcolo spread iniziale per tutti i nodi (può richiedere tempo)...
Candidati validi (out-degree > 0): 16522
  Analizzati 0/16522...
  Analizzati 1000/16522...
  Analizzati 2000/16522...
  Analizzati 3000/16522...
  Analizzati 4000/16522...
  Analizzati 5000/16522...
  Analizzati 6000/16522...
  Analizzati 7000/16522...
  Analizzati 8000/16522...
  Analizzati 9000/16522...
  Analizzati 10000/16522...
  Analizzati 11000/16522...
  Analizzati 12000/16522...
  Analizzati 13000/16522...
  Analizzati 14000/16522...
  Analizzati 15000/16522...
  Analizzati 16000/16522...
Seme 1 trovato: 55125380 (Spread: 162.08)
Seme 2 trovato: 138224980 (Gain: 95.98)
Seme 3 trovato: 53097946 (Gain: 77.32)
Seme 4 trovato: 76860837 (Gain: 56.32)
Seme 5 trovato: 86438122 (Gain: 47.50)
CELF completato in 3.00 secondi.

Confronto Strategie IM (Monte Carlo)...
                    Strategy                                          Seeds  Spread
0      CELF (Greedy Approx)  [55125380, 138224980, 53097946, 76860837, 8643...  439.20
1     Top Out-Degree (Hubs)  [55125380, 138224980, 53097946, 76860837, 8643...  437.86
2  Top PageRank (Authority)  [55125380, 162622596, 138224980, 86438122, 530...  390.24
Analisi Incongruenze Rating...
Trovate 39 incongruenze (scarto >= 2) su 8721 coppie valide.

Salvataggio GEXF...

==================================================
✅ TUTTO COMPLETATO.
==================================================
```

Figura 3.1: Log di fine esecuzione dello script `asnm_analysis.py`

Conclusioni del Capitolo: L'analisi dinamica ha confermato che la struttura commerciale di Vinted influisce fortemente sulla diffusione della fiducia:

- La topologia Hub-and-Spoke limita la propagazione spontanea tra acquirenti (clustering basso), rendendo la scelta dei Seed cruciale.

- Affidarsi alla sola autorità (PageRank) è sub-ottimale per il marketing virale in questo contesto (-15% di efficacia rispetto a CELF).

- CELF si conferma l'approccio più robusto, identificando un set di 5 utenti capaci di attivare, da soli, una porzione significativa della rete osservata.

Capitolo 4

Visualizzazione del Grafo e Interpretazione dei Risultati

L'analisi numerica, sebbene fondamentale, non riesce sempre a restituire la complessità topologica di una rete sociale. Per questo motivo, i risultati quantitativi ottenuti nel capitolo precedente sono stati affiancati da un'analisi qualitativa basata sulla visualizzazione grafica. Il grafo finale (`vinted_graph_FINAL.gexf`) è stato importato nel software **Gephi**, dove sono state applicate tecniche di spazializzazione (Layout) e filtraggio visivo per rendere leggibile la struttura delle comunità e verificare empiricamente la presenza di fenomeni di aggregazione (Cluster) e di connessione (Brokerage).

4.1 Algoritmo di Layout e Spatialization

La disposizione spaziale dei nodi non è casuale, ma è il risultato dell'applicazione di un algoritmo *force-directed*. È stato scelto il layout **ForceAtlas 2**, noto per la sua capacità di gestire reti di medie-grandi dimensioni e per l'efficacia nel separare visivamente le comunità densamente connesse.
I parametri di configurazione sono stati ottimizzati per favorire la leggibilità della struttura modulare:

- **Scaling = 5.0:** Aumenta la repulsione tra i nodi, "esplodendo" il grafo per rendere visibili i gap strutturali tra le diverse comunità tematiche.

- **Gravity = 1.0:** Mantiene una coesione globale, impedendo ai nodi isolati di disperdersi eccessivamente nello spazio grafico.
- **Prevent Overlap = ON:** Impedisce la sovrapposizione grafica dei nodi, garantendo che ogni singolo utente (nodo) sia distinguibile.
- **Dissuade Hubs = ON:** Introduce una forza repulsiva addizionale per i nodi con alto grado. Questo parametro è stato cruciale per evitare che gli Hub (venditori molto attivi) collassassero al centro del grafo coprendo le strutture periferiche, spingendoli invece al centro delle loro rispettive comunità di appartenenza.

Il risultato di questa configurazione è una distribuzione a "galassia" in cui i cluster semantici appaiono come isole ben distinte, collegate tra loro da sottili filamenti di connessione (i venditori ibridi).

4.2 Analisi Semantica tramite Colorazione (Partition)

Per verificare la corrispondenza tra la topologia della rete (calcolata matematicamente da Louvain) e il contenuto semantico (i tag estratti dallo scraping), è stata applicata una colorazione basata sui metadati.
Nel pannello *Appearance* → *Partition*, i nodi sono stati colorati in base all'attributo `Community_Label`, generato automaticamente dallo script di analisi Python. Questa visualizzazione ha permesso di validare due ipotesi fondamentali:

1. **Omogeneità Interna:** I cluster spaziali formati da ForceAtlas2 presentano una colorazione uniforme. Ad esempio, la zona del grafo densamente popolata da nodi di colore distintivo corrisponde interamente alla community *TRADING CARD GAMES*, confermando che gli scambi avvengono prevalentemente tra utenti con lo stesso interesse verticale (Omofilia).
2. **Zone di Confine (Mix):** Le aree interstiziali tra un cluster e l'altro mostrano nodi con etichette *MIX* (es. *MIX: FUMETTI & ACTION*),

evidenziando visivamente i punti di contatto dove operano i venditori Multi-Category.

4.3 Gerarchia dei Nodi e Centralità (Ranking)

Per evidenziare la gerarchia sociale all'interno del marketplace, la dimensione dei nodi è stata correlata alle metriche di centralità calcolate. Nel pannello *Appearance* → *Size*, è stato applicato un ranking basato sulla **Betweenness Centrality**.

Questa scelta visuale è strategica: dimensionare i nodi per Betweenness (piuttosto che per semplice Grado) fa emergere visivamente i **Broker**, ovvero quegli utenti che, pur non avendo necessariamente il maggior numero di vendite in assoluto, occupano posizioni critiche di connessione tra sottogruppi diversi. Il grafo risultante mostra una struttura *Core-Periphery*: al centro delle "isole" e lungo i "ponti" risiedono i nodi giganti (Broker/Hub), mentre la periferia è popolata da una vasta nube di nodi piccoli (utenti occasionali o specialisti isolati).

4.4 Gestione delle Etichette e Leggibilità

Data la densità della rete (oltre 16.000 nodi), la visualizzazione simultanea di tutte le etichette testuali avrebbe reso il grafo illeggibile ("effetto nuvola nera"). È stata quindi adottata una strategia di *Selective Labeling* basata sulle dimensioni.

Nel pannello *Anteprima (Preview)* di Gephi, è stata attivata l'opzione **Proportional Size** per le etichette dei nodi. Questo filtro dinamico renderizza il testo dell'etichetta (il nome della Community, es. "06 - MODA & LUSSO") con una dimensione proporzionale alla grandezza del nodo stesso. Impostando una soglia minima di font, le etichette dei nodi irrilevanti diventano invisibili, mentre i nomi delle categorie dominanti appaiono chiaramente solo sopra i grandi Hub centrali. Questo accorgimento ha permesso di generare una "mappa geografica" del mercato Vinted pulita e auto-esplicativa.

4.5 Visualizzazione Grafica

Di seguito viene riportata un'immagine di come appare visivamente il grafo:

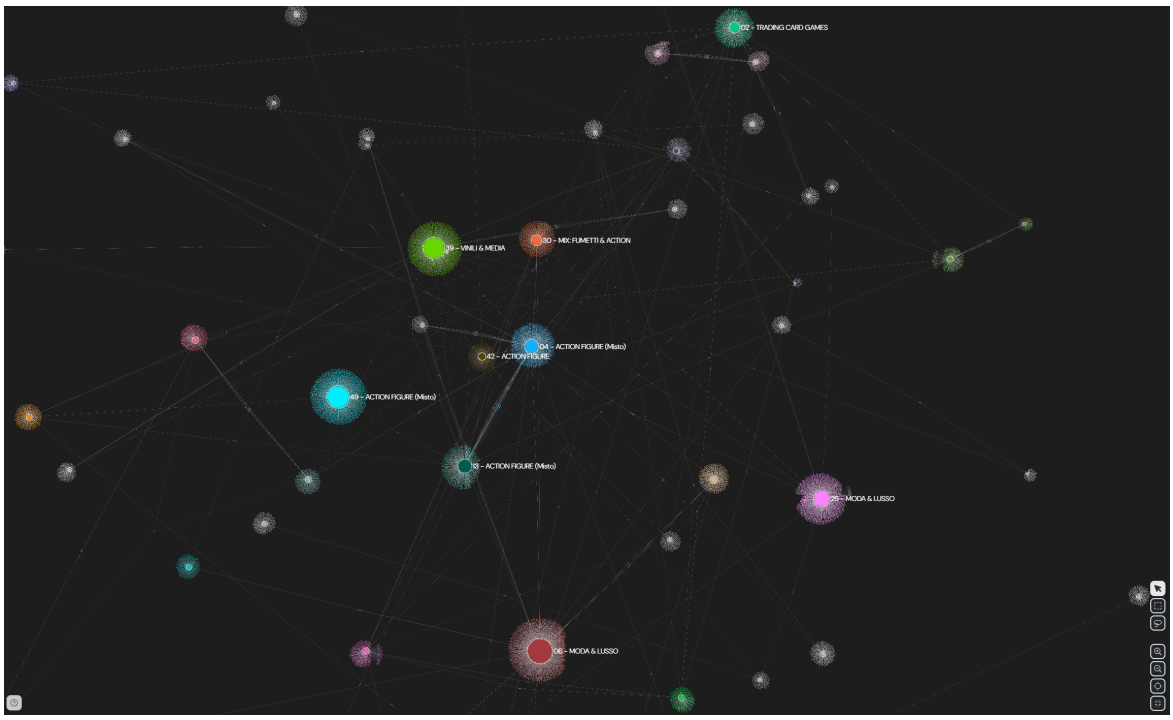

Figura 4.1: Snapshot dell'intera rete Vinted

Per una più approfondita visualizzazione grafica si rimanda all'uso di Gephi Lite disponibile al seguente link: https://lite.gephi.org/v1.0.1, caricando il gexf del grafo

Nella sezione "Data" si possono ordinare gli utenti per degree, copiare l'ID dei seeds e cercarli su Vinted. Per fare ciò basta cercare un utente qualsiasi su Vinted e sostituire nell'URL il suo ID con l'ID di interesse.

4.6 Sintesi delle Osservazioni Qualitative

L'ispezione visiva del grafo conferma le metriche quantitative e offre ulteriori insight sulla natura delle interazioni su Vinted:

- **Segregazione degli Interessi:** Le community relative a *Moda & Lusso* e *Carte TCG* appaiono topologicamente molto distanti, quasi isolate, confermando che raramente lo stesso utente opera contemporaneamente in questi due mercati antipodi.

- **Il Ruolo dei Ponti:** Le community di *Fumetti*, *Action Figure* e *Videogiochi* appaiono invece più vicine e interconnesse, formando una macroarea "Nerd Culture". Qui si osserva la maggiore concentrazione di nodi "Ponte" (alta Betweenness), che agiscono da collante per l'intero ecosistema.

- **Struttura Hub-and-Spoke:** La visualizzazione conferma la bassa densità globale: le community non sono "clique" perfette (tutti connessi con tutti), ma assomigliano a sistemi solari, con grandi venditori al centro che attirano orbite di acquirenti che non interagiscono tra loro. Gli Hub semantici (es. grandi venditori di Carte) fungono da centri di gravità ; la loro rimozione causerebbe probabilmente la frammentazione della community in componenti disconnesse, confermando la loro importanza strutturale oltre che commerciale.

Capitolo 5

Conclusioni

Il lavoro svolto ha permesso di trasformare Vinted, originariamente concepito come un semplice marketplace C2C, in un *social network analizzabile* secondo i paradigmi della Social Network Analysis. Attraverso una pipeline completa — dal recupero dei dati tramite crawling controllato alla modellazione del grafo orientato e pesato, fino alle simulazioni dinamiche di diffusione — è stato possibile osservare fenomeni complessi legati alla reputazione, alla segregazione delle community e al ruolo strategico degli utenti influenti.

Risultati principali

Il progetto ha portato a tre risultati chiave che validano l'approccio metodologico adottato:

1. **Validazione della natura sociale (Reciprocità)**. La ricostruzione dei feedback bidirezionali ha svelato una rete caratterizzata da un coefficiente di **Reciprocità elevato** (≈ 0.67). Questo dato quantitativo è la prova definitiva che Vinted non opera solo come un canale di vendita unidirezionale, ma si fonda su meccanismi di fiducia mutua tipici delle reti sociali, distinguendosi nettamente dai classici e-commerce B2C.

2. **Segregazione tematica marcata (Modularità)**. L'applicazione dell'algoritmo di Louvain, combinata con la strategia di *Smart Naming*, ha prodotto una partizione con una **Modularità eccezionalmente alta (0.94)**. Questo conferma l'esistenza di "isole" commerciali ben definite

(es. *TRADING CARD GAMES*, *MODA & LUSSO*) e scarsamente permeabili, dove le transazioni tendono a rimanere confinate all'interno di nicchie verticali.

3. **Supremazia dei Broker nell'Influenza**. Le simulazioni di *Influence Maximization* hanno dimostrato la superiorità dell'algoritmo CELF rispetto alle euristiche basate sulla centralità pura (Top Degree/PageRank). I nodi selezionati da CELF hanno massimizzato lo spread dell'informazione minimizzando la sovrapposizione. Ciò dimostra che per diffondere un trend su Vinted non serve colpire i "venditori più grossi" (spesso chiusi nella loro nicchia), ma i nodi **Multi-Category** (Broker) posizionati strategicamente ai confini tra le community.

Risposta alla domanda di ricerca

La domanda iniziale chiedeva:

> *La reputazione in una rete di transazioni si diffonde più efficacemente all'interno delle community o attraverso i nodi ponte che le collegano?*

Dalle analisi emerge una dualità funzionale:

- **All'interno delle community (Intra-cluster)**, la diffusione è rapida e capillare grazie all'alta densità locale e alla fiducia reciproca.
- **Tra le community (Inter-cluster)**, la propagazione dipende quasi esclusivamente dai nodi ponte identificati come **Multi-Category Mix**. Senza questi attori ibridi, la fiducia rimarrebbe segregata nelle singole nicchie. Pertanto, per una diffusione globale sulla piattaforma, i nodi ponte sono i veri vettori critici.

Limiti e sviluppi futuri

Nonostante i risultati promettenti, il progetto presenta margini di miglioramento che aprono a sviluppi futuri:

- **Espansione del dataset**: L'assenza di API ufficiali e i limiti di rate-limiting hanno ristretto il campionamento. Uno scraping distribuito su più nodi permetterebbe di coprire un numero maggiore di utenti e categorie merceologiche.

- **Arricchimento semantico con NLP**: L'attuale sistema di tagging basato su keyword potrebbe essere esteso integrando *embedding* testuali (es. BERT) sulle descrizioni degli articoli, migliorando la classificazione dei nodi ibridi.

- **Analisi temporale**: Lo studio dell'evoluzione della rete nel tempo (Dynamic Networks) permetterebbe di osservare come nascono le community e come variano i pattern reputazionali a seguito di frodi o cambiamenti nelle policy della piattaforma.

Considerazioni finali

Il lavoro dimostra come un ecosistema commerciale possa essere efficacemente interpretato come un sistema sociale complesso. Analizzare Vinted come una rete di interazioni, piuttosto che come un semplice database di prodotti, ha permesso di cogliere fenomeni invisibili all'utente finale: l'esistenza di comunità latenti, l'importanza strutturale degli utenti ponte e le dinamiche epidemiche della fiducia.
In conclusione, la combinazione di tecniche di *crawling* controllato, modellazione dei dati, SNA e simulazione dinamica ha fornito una comprensione profonda della struttura funzionale di Vinted, evidenziando come le interazioni sociali siano il vero motore che sostiene l'economia della piattaforma.