COMMUNITY DETECTION

Su generi musicali di utenti ascoltatori di LastFM

Dettagli Documento

Gruppo	**Membri**
Rido Poco	Alessandro Iori, Daniel Morales
Revisione	**Descrizione**
1.0 del 12 Luglio 2017	Secondo progetto Big Data
Emissione	**Validazione**
Daniel Morales, Alessandro Iori	

Introduzione

Scopo del documento

Lo scopo che si prefigge questo documento è mostrare la progettazioni e l'analisi del progetto di Community Detection per il corso di Laurea Magistrale di Big Data dell'Università degli Studi Roma Tre, proposto dall'Ing. Donatella Firmani collaboratrice del docente del corso prof. Riccardo Torlone.

• Sito web del corso: Big Data @ Roma Tre
• Slide di introduzione e proposta di progetto dell'Ing. Donatella Firmani: A5-community

Il codice del progetto realizzato è disponibile al seguente link:
• (repository progetto) https://github.com/alessandroiori/community-detection-lastfm

Che cos'è la Community Detection?

"*Community detection is key to understanding the structure of complex networks, and ultimately extracting useful information from them*" - MIT Edu

La community detection è lo studio di reti complesse, all'interno delle quali è possibile individuare gruppi in cui la membership non è esplicitamente data.

Ha rilevante importanza in molte discipline della scienza dove il l'informazione è spesso rappresentata sotto forma di grafo.

Le applicazioni sono diverse: dalla salute alla geografia, dalle interazioni umane all'economia.

Obiettivo

Il nostro obiettivo è quello di individuare comunità di utenti ascoltatori di lastFM sulla base dei gusti musicali.
LastFM è un social network e portale per l'ascolto di musica, sito "progetto" di Audioscrobbler. Gli utenti possono ascoltare musica, aggiungere tag descrittivi a ciascun artista, seguire artisti e altri utenti.

Dataset di riferimento

Il dataset di riferimento che stiamo utilizzando è fornito da GroupLens, gruppo di ricerca dell'Università del Minnesota.
I dati contengono informazioni riguardo il social networking, tagging e ascolti di artisti musicali da un insieme di 2K utenti di LastFM. Il dataset è stato rilasciato per il 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems (HetRec 2011).
In particolare questo dataset contiene:

- 2000 utenti
- 18k artisti
- 13k relazioni amicizia tra utenti
- 93k relazioni di ascolto utente-artista
- 186k assegnazioni di tag agli artisti

Di seguito il dettaglio:

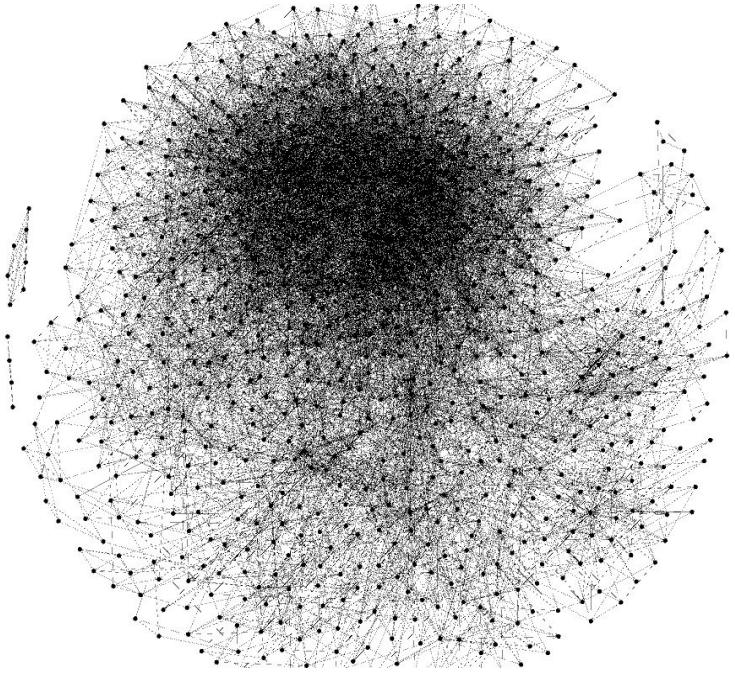
```
  1892 users
 17632 artists

 12717 bi-directional user friend relations, i.e. 25434 (user_i, user_j) pairs
       avg. 13.443 friend relations per user

 92834 user-listened artist relations, i.e. tuples [user, artist, listeningCount]
       avg. 49.067 artists most listened by each user
       avg. 5.265 users who listened each artist

 11946 tags

186479 tag assignments (tas), i.e. tuples [user, tag, artist]
       avg. 98.562 tas per user
       avg. 14.891 tas per artist
       avg. 18.930 distinct tags used by each user
       avg. 8.764 distinct tags used for each artist
```

Descrizione progetto

Oltre all'obiettivo principale che ci siamo posti di soddisfare, in maniera trasversale ci siamo proposti di studiare e confrontare diverse tecniche di Community Detection. Il progetto inoltre è realizzato grazie all'utilizzo di opportune tecnologie Big Data.

Inizialmente per riuscire ad individuare comunità di utenti con gli stessi gusti musicali, ci siamo basati sulle relazioni di amicizia che sussistono fra questi.
L'assunzione iniziale era quella che due utenti amici (che si seguono) avessero gli stessi gusti musicali, quindi abbiamo utilizzato l'algoritmo di community detection: **Clique** per riuscire a far emergere queste comunità.
Il risultato ottenuto dall'elaborazione di clique sulle relazioni di amicizia non ha prodotto nessun valore informativo, in quanto dal grafico non emergono le comunità in maniera evidente.

Di seguito il grafico:



Abbandonata la strada delle relazioni di amicizia abbiamo deciso di modellare le relazioni tra utenti in maniera implicita.
L'idea è stata quella di creare relazioni fra utenti che hanno un **comportamento** simile nel social network. Un dato rilevante riguardo al comportamento degli utenti di cui siamo in possesso è il numero di ascolti di un utente per artista.
Nei prossimi capitoli saranno discusse le metodologie e tecniche che ci hanno permesso di identificare le comunità per gusti musicali.

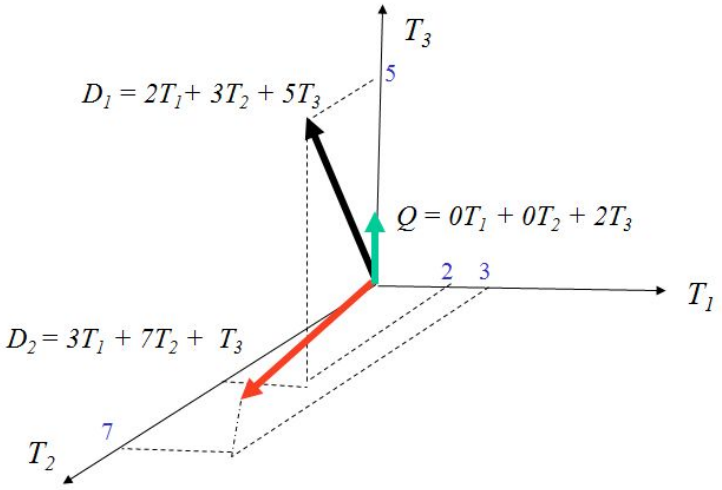
Analisi

In questo capitolo saranno discusse le tecniche che ci hanno permesso di ottenere il grafo delle relazioni fra utenti e gli algoritmi per la rilevazione di comunità.

Modellazione

Dalle analisi preliminari è emerso che le relazioni di amicizia non consentono di identificare comunità coerenti.
Il passo successivo è stato quello di mettere in relazione utenti sulla base del comportamento nei social network. Per farlo abbiamo utilizzato il dataset *user-listening.dat*, contenente gli ascolti di ciascun utente nei confronti degli artisti.

L'idea è stata quella di modellare un utente come un vettore nello spazio degli artisti, ovvero, attraverso una feature vector.
Una strategia spesso usata nell'Information Retrieval per estrarre documenti rilevanti per la query sottomessa dall'utente.
Questa rappresentazione permette, quindi, di trovare gruppi di utenti molto simili sulla base degli ascolti.



Pairwise Similarity

Per modellare la forza delle relazioni fra utenti abbiamo usato la misura di pairwise-similarity anche nota come cosine-similarity con la quale è stato possibile estrarre la similarità fra utenti.
La coseno similarità: *"[...] è una tecnica euristica per la misurazione della similitudine tra due vettori effettuata calcolando il coseno tra di loro"* - cit. Wikipedia.
In base alla definizione del coseno, dati due vettori si otterrà sempre un valore di similitudine compreso tra -1 e +1, dove -1 indica una corrispondenza esatta ma opposta (ossia un vettore contiene l'opposto dei valori presenti nell'altro) e +1 indica due vettori uguali.
Nel nostro caso, poiché gli ascolti degli artisti sono sempre valori positivi, si otterranno valori che vanno da 0 a +1, dove +1 sono le coppie di utenti che ascoltano esattamente gli stessi artisti e 0 quelle che non ascoltano gli stessi artisti.

Grafo di Similarità

Una volta ottenuto il grado di similarità per ogni coppia di utenti abbiamo costruito il grafo andando a eliminare tutte le relazioni con una soglia di similarità inferiore ad un certo valore.
In particolare la threshold è stata individuata dall'analisi qualitativa dei grafi costruiti e con l'ausilio dello strumento di esplorazione e visualizzazione **Gephi** che ci ha permesso di determinare la soglia a 0.6.

Community Detection

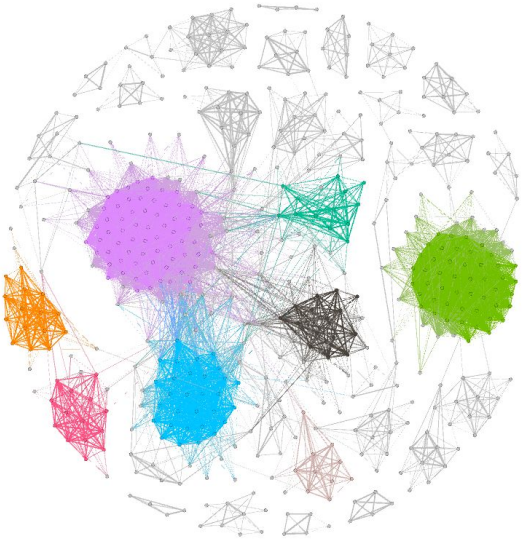
In questa fase del progetto abbiamo utilizzato tre algoritmi noti per rilevare le comunità.
Di seguito la descrizione degli algoritmi utilizzati per individuare comunità di utenti e il loro output.

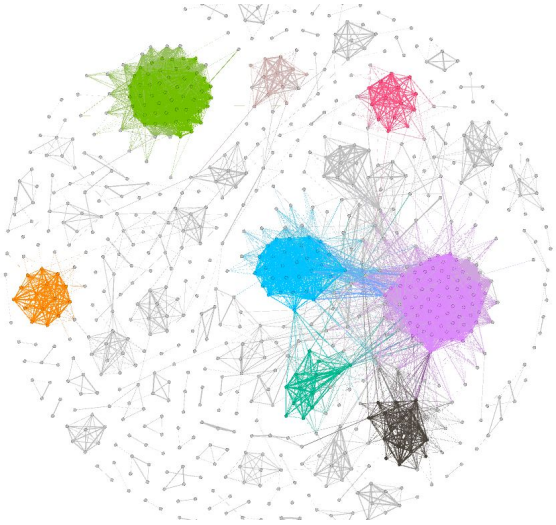
Clique

Un algoritmo classico nello studio delle comunità, di seguito la sua definizione.

"*In teoria dei grafi, una cricca (o clique) è un insieme V di vertici in un grafo non orientato G, tale che, per ogni coppia di vertici in V, esiste un arco che li collega.*" - Wikipedia

L'esecuzione dell'algoritmo ha prodotto la visualizzazione a destra.





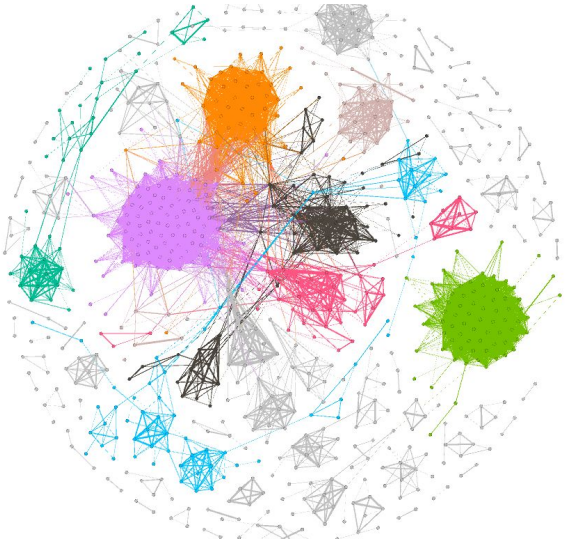
K-plex

"*A k-plex is a maximal subgraph with the following property: each vertex of the induced subgraph is connected to at least n-k other vertices, where n is the number of vertices in the induced subgraph*" - Analytictech

L'esecuzione dell'algoritmo ha prodotto la visualizzazione a sinistra.

Louvain Method

"*The Louvain Method for community detection is a method to extract communities from large networks. [...]The inspiration for this method of community detection is the optimization of Modularity as the algorithm progresses. Modularity is a scale value between -1 and 1 that measures the density of edges inside communities to edges outside communities.*" - Wikipedia



Progettazione

In questo capitolo saranno discussi gli strumenti e le tecniche che ci hanno permesso di ottenere il grafo delle relazioni e le comunità fra utenti.

Tecnologie utilizzate

E' stato deciso di realizzare un sistema poliglotta dal punto di vista delle tecnologie e dei linguaggi in base all'esigenza di elaborazione del dato. Elenco delle tecnologie e linguaggi rilevanti utilizzati per la realizzazione del progetto:

- Python3
- NetworkX
- Java
- MLlib
- Spark
- Cassandra
- Neo4j 3.0
- Gephi
- Docker
- Amazon AWS
    - EC2 Compute Optimized
    - S3 Bucket

Fasi del Progetto

Il progetto si suddivide in tre fasi propedeutiche principali:

1. Individuazione delle relazioni tra utenti;
2. Individuazione delle community;
3. Modellazione e popolamento graph database Neo4J.

Individuazione delle relazioni

In prima istanza sono stati caricati i dati (vedi Dataset di Riferimento) su Cassandra, un DBMS distribuito e open source, attraverso degli script realizzati ad-hoc.

Successivamente è stata realizzata un'applicazione standalone per calcolare la similarità fra utenti. L'applicazione è stata realizzata con il framework open source per il calcolo distribuito Spark che ci ha permesso di calcolare in maniera efficace la coseno similarità per ciascuna coppia di utenti, come descritto nel paragrafo di Modellazione.

Infatti, Spark offre la libreria MLlib che mette a disposizione diverse funzioni per il Machine Learning tra cui appunto la Cosine Similarity.
Il nostro dataset, ormai caricato su Cassandra, *user-listenings.dat* rappresenta tutti gli ascolti degli utenti nei confronti degli artisti. Poiché la rappresentazione di questa matrice è sparsa abbiamo sfruttato le strutture dati distribuite (Sparse Vector) offerte da MLlib per poter caricare tali informazioni e calcolare la coseno similarità.
Una volta calcolata questa metrica, le relazioni in output sono state caricate su Cassandra.

Quest'ultimo accorgimento ci ha permesso di poter ricavare le coppie di utenti a diversi livelli di similarità semplicemente attraverso una query SQL e di caricare questi dati su un file csv.
L'applicazione è stata fatta eseguire in ambiente locale su contenitori Docker per definire un ambiente di esecuzione isolato e autocontenuto. In particolare due contenitori Docker, uno per Cassandra e l'altro per Spark.

Codice disponibile al seguente path della repository del progetto:
/community-detection-lastfm/code/**relationship-detection**/graph-core/

Individuazione delle community

La rilevazione delle comunità ha previsto sia l'utilizzo degli algoritmi di community detection, già descritti nel paragrafo di Community Detection, sia la visualizzazione dell'output di questi.

Ciascun algoritmo è stato implementato in un modulo separato che ha la responsabilità di produrre i file per la visualizzazione delle comunità sullo strumento Gephi e per memorizzare l'insieme delle comunità.
In particolare poiché Gephi consente di codificare alcune etichette dei nodi e/o degli archi attraverso il colore, abbiamo fatto in modo che ciascun modulo rappresentasse l'informazione della comunità in cui ciascun nodo partecipa. Questo è stato realizzato attraverso la creazione di file in formato gefx (standard Gephi) in cui ciascun nodo ha una label per rappresentare l'identificatore della sua comunità.

I moduli sono stati realizzati in linguaggio Python, in particolare l'estrazione delle clique e louvain method sono stati realizzati attraverso la libreria **NetworkX** che rappresenta la soluzione più efficace.
Mentre K-Plex è stato realizzato attraverso la libreria **snap.py** offerta da Standford university.

In tutti e tre i casi le applicazioni sono state fatte eseguire su delle istanze separate di **EC2 Compute Optimized** (c.large: 2 vCPU, 3.75 GB) di Amazon Web Services, a causa dei lunghi tempi di esecuzione necessari per l'elaborazione di algoritmi NP-hard.
L'esecuzione degli algoritmi è stata automatizzata in modo da prevedere lo spegnimento e la sincronizzazione dei file generati su un **bucket S3** (sempre AWS), in modo tale da farla eseguire in autonomia durante la notte e senza costringerci a monitorarne l'esecuzione.

I tempi relativi all'esecuzione dei diversi algoritmi sono elencati di seguito:

K-Plex	~5h 31m
Clique	~5s
Louvain method	~3s

Al termine i dati sono stati raccolti e caricati nel Database a grafo **Neo4J** per l'analisi dei risultati e la validazione.

Codice disponibile al seguente path della repository del progetto:
/community-detection-lastfm/code/**community-detection**/

Neo4J: Modellazione e popolamento del database

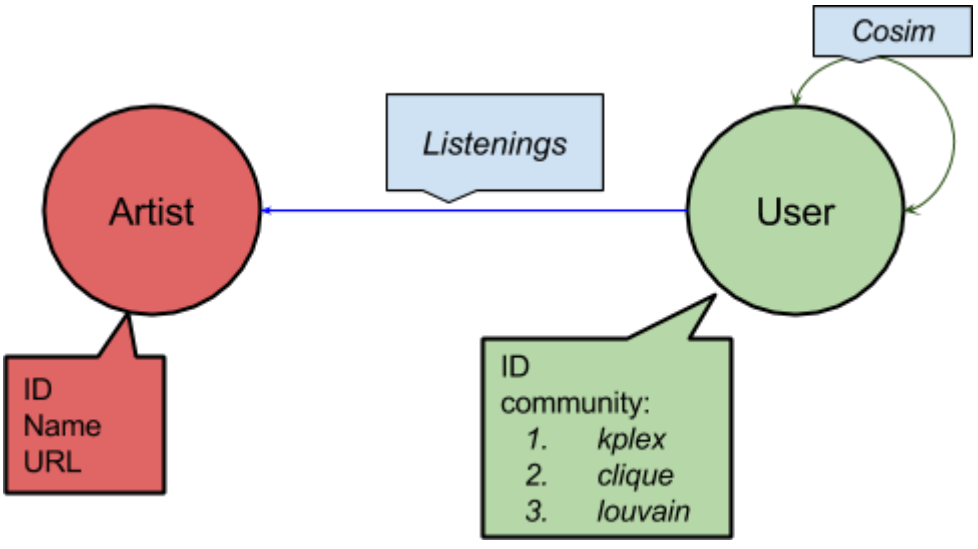
Motivati dalla complessità delle relazioni e dalla struttura intrinseca del dataset, abbiamo deciso di utilizzare un graph database per la memorizzazione dei dati, in particolare **Neo4j**, software per basi di dati a grafo open source, interamente sviluppato in Java.
Il database viene eseguito in locale all'interno di un container **Docker** con il seguente comando:

```
docker run \
--publish=7474:7474 --publish=7687:7687 \
--volume=$HOME/neo4j/data:/data \
--volume=$HOME/neo4j/logs:/logs \
--volume=$HOME/neo4j/files:/var/lib/neo4j/import/files \
neo4j:3.0
```

Il servizio mette a disposizione sulla porta 7474 del localhost una dashboard che permette di visualizzare in maniera interattiva l'output delle query sottomesse. Le query sono scritte in Cypher, linguaggio dichiarativo per grafi, simile all'SQL, che consente di effettuare efficienti query sui graph database.

Modellazione

Per favorire l'analisi comparativa dei risultati prodotti dai vari algoritmi è stato necessario realizzare una modellazione del dominio di interesse.



Sono state individuate due entità principali e due tipi di relazioni:
- Entitità:

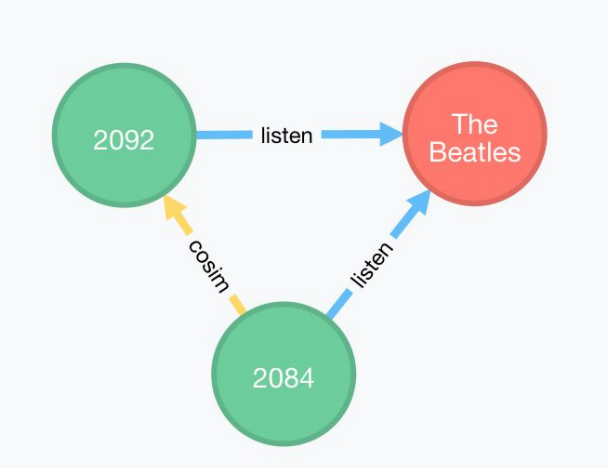
- **Users**:
  - **ID**: Identificatore dell'utente
  - **community**:
    - **k-plex**: lista di identificatori di plessi a cui partecipa l'utente;
    - **clique**: lista di identificatori di clique a cui partecipa l'utente;
    - **louvain**: valore che indica la singola comunità louvain a cui partecipa l'utente.
- **Artists**:
  - **ID**: Identificatore dell'Artista;
  - **Name**: nome dell'Artista;
  - **URL**: URL del profilo dell'Artista;

- Relazioni:

- **Listening**:
  Rappresenta gli ascolti di un utente per un certo artista.
  - weight: Identificatore del numero di ascolti.

- **Cosim**:
  Rappresenta la coseno similarità tra due nodi
  - weight: Valore decimale compreso tra 0.6 e 1 che identifica quanto due utenti sono simili. Maggiore è il valore, più due utenti sono simili tra di loro.

Esempi di entità e relazioni su Neo4j:



Artist

name	The Beatles
artistId	227
url	http://www.last.fm/music/The+Beatles

User

kplexes	[346, 625, 626]
cliques	[142, 332]
louvains	[8]
userId	2084

Cosim

weight	0.8173175240835568

Listening

weight	891

Popolamento del database

Il popolamento del database è avvenuto mediante query Cypher immerse all'interno di script ad-hoc in linguaggio Python. In particolare il flusso di caricamento è avvenuto come segue:

- Il flusso ha avuto inizio caricando gli Utenti, gli Artisti e le relazioni di Listening tra il primo e il secondo.
- Successivamente sono state caricate le relazioni della coseno similarità tra coppie di utenti.
- Infine abbiamo assegnato a ciascun nodo utente, l'identificativo della comunità dello specifico algoritmo di community detection (clique, k-plex, louvain) a cui appartiene.

Per ottimizzare le query su Utenti e Artisti, sono stati creati due indici sulla proprietà id delle due entità con i seguenti comandi in linguaggio Cypher:

```
CREATE INDEX ON :User(userId)
CREATE INDEX ON :Artist(artistId)
```

Di seguito porzioni di alcuni script utilizzati per effettuare il popolamento del database.

Autenticazione a Neo4j:

```
driver = GraphDatabase.driver(uri_neo4j_db, auth=(user_auth, pw_auth))
session = driver.session()
```

Nella seguente porzione di script vengono creati i nodi Utente, Artista e la relazione di listening tra di essi:

```
q =
"MERGE (u:User {userId: {user_id}}) \
MERGE (a:Artist {artistId: {artist_id}}) \
MERGE (u)-[r:listen {weight: {weight}}]->(a) \
RETURN u, a, r"

result = session.run(q, {"user_id": user_id, "artist_id": artist_id,
"weight": weight})
```

Porzione di script che consente di assegnare agli utenti l'appartenenza a community identificate dall'algoritmo k-plex:

File condiviso con il contenitore Docker contenente righe le quali identificano community, composte a loro volta da ID di utenti che partecipano alla comunità:

```
kplex_file = "file:///files/kplex/cosim.2plexes"
```

La seguente query in Cypher consente di :

- iterare su ciascuna riga del file;
- per ciascun ID dell'utente contenuto nella riga, ottenere il nodo corrispondente;
- per ciascun nodo se non esiste, viene assegnata la proprietà kplexes;
- aggiungere l'identificativo della rispettiva riga (quindi relativa comunità) all'interno della proprietà kplexes;

```
q_kplex =
'LOAD CSV FROM {file} AS line FIELDTERMINATOR ":" ' \
'WITH SPLIT(line[1], " ") AS usersId, SPLIT(line[0], " ") AS row_num ' \
'FOREACH ' \
'( uId IN usersId | ' \
'MERGE (u:User {userId: uId}) ' \
'SET u.kplexes = coalesce(u.kplexes ,[]) + row_num ' \
')' \
'RETURN usersId, row_num'
```

Esecuzione della query su Neo4j:

```
result_kplex = session.run(q_kplex, {"file": kplex15_file})
```

Il resto del database è stato popolato con script e query simili a quelle descritte sopra.

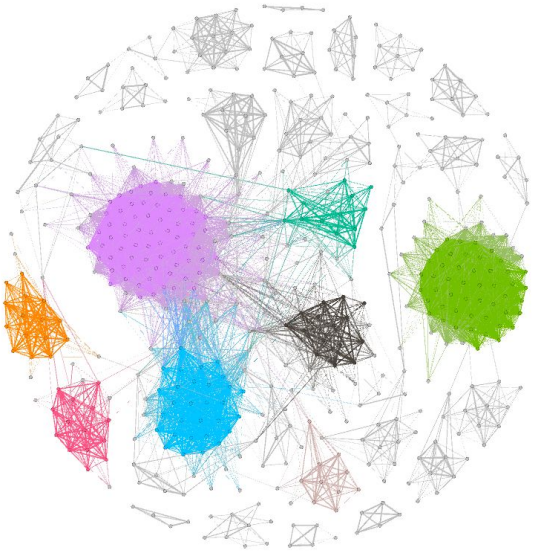
E' possibile trovare il codice relativo a Neo4j all'interno del repository al seguente path: /community-detection-lastfm/code/**community-detection**/

Analisi dei Risultati

In questo capitolo analizzeremo quanto prodotto dagli algoritmi in maniera da validare i risultati ottenuti.

Validazione

In questo paragrafo verificheremo che le comunità prodotte dai diversi algoritmi riflettano effettivamente i gusti musicali degli utenti. Le comunità selezionate visibilmente su Gephi, sono state analizzate attraverso delle query Cypher su Neo4J.



Clique

Per verificare la correttezza dell'algoritmo di community detection clique, abbiamo selezionato un campione di comunità, in particolare quelle rappresentate con il colore verde e rosa nell'immagine a destra.

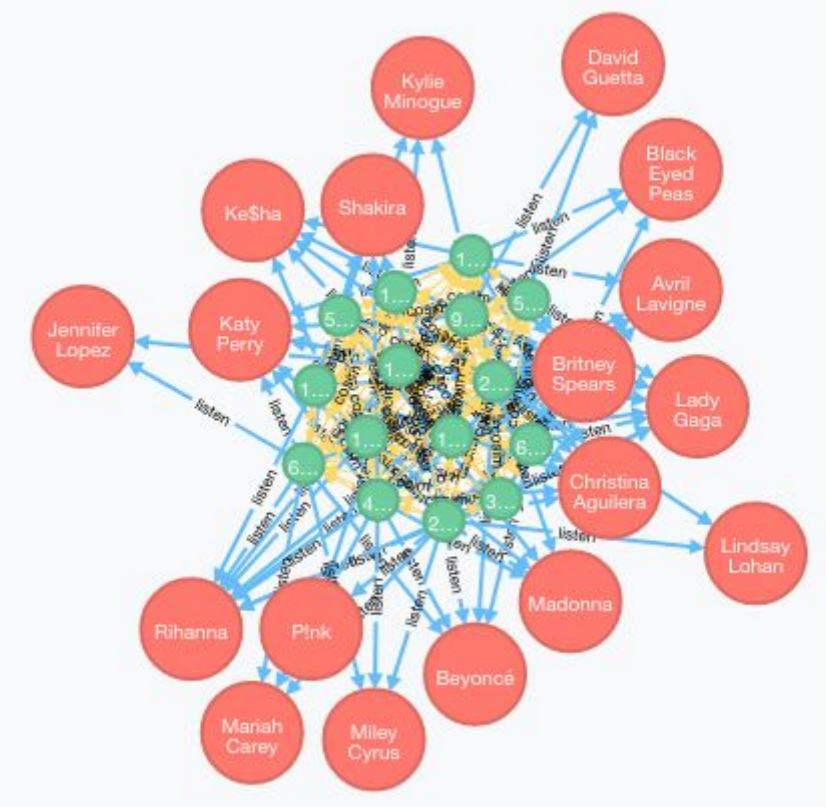
La comunità verde che corrisponde all'Id 30 è stata analizzata con la seguente query:

```
MATCH p=(u1:User)-[r1:cosim]-(u2:User)
MATCH q=(u1:User)-[r2:listen]->(a:Artist)
MATCH f=(u2:User)-[r3:listen]->(a:Artist)
WHERE "30" IN u1.cs AND "30" IN u2.cs
AND toInt(r2.weight) >= 1000 AND toInt(r3.weight) >= 1000
return p, q, f
```

L'esecuzione della query ha prodotto il grafo a sinitra.

Dall'analisi del grafo emerge che la comunità verde ascolta artisti del genere prevalentemente Rock.

Mentre dall'analisi della comunità rosa, mostrata nell'immagine in basso, emerge che gli ascoltatori sono interessati al genere Pop.

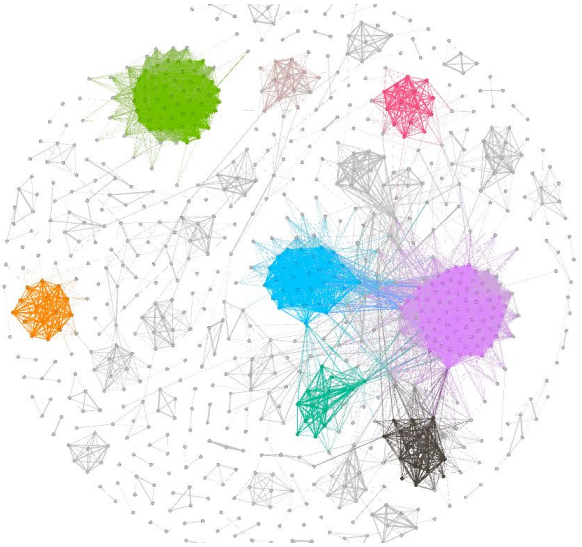


Query Cypher utilizzata per ottenere il grafico:

```
MATCH p=(u1:User)-[r1:cosim]-(u2:User)
MATCH q=(u1:User)-[r2:listen]->(a:Artist)
MATCH f=(u2:User)-[r3:listen]->(a:Artist)
WHERE "1" IN u1.cs AND "1" IN u2.cs AND toInt(r2.weight) >= 1000 AND
toInt(r3.weight) >= 1000
return p, q, f
```

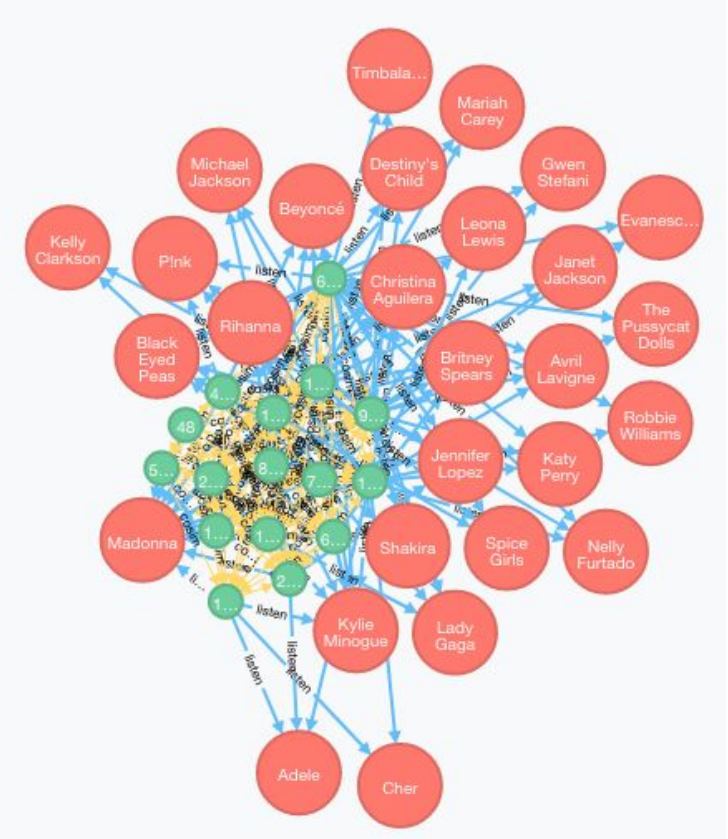
K-plex

Di seguito alcune analisi relative ai gusti musicali per le comunità rilevate attraverso l'algoritmo del k-plesso. Per validare tale algoritmo abbiamo selezionato delle comunità da Ghepi e visualizzate su Neo4j. In queste comunità come per Clique possiamo notare che viene rappresentata un singolo genere musicale ma con più varietà e meno staticità di clique.



Grafo Gephi di tutte le comunità individuate con K-plex nell'immagine a destra.

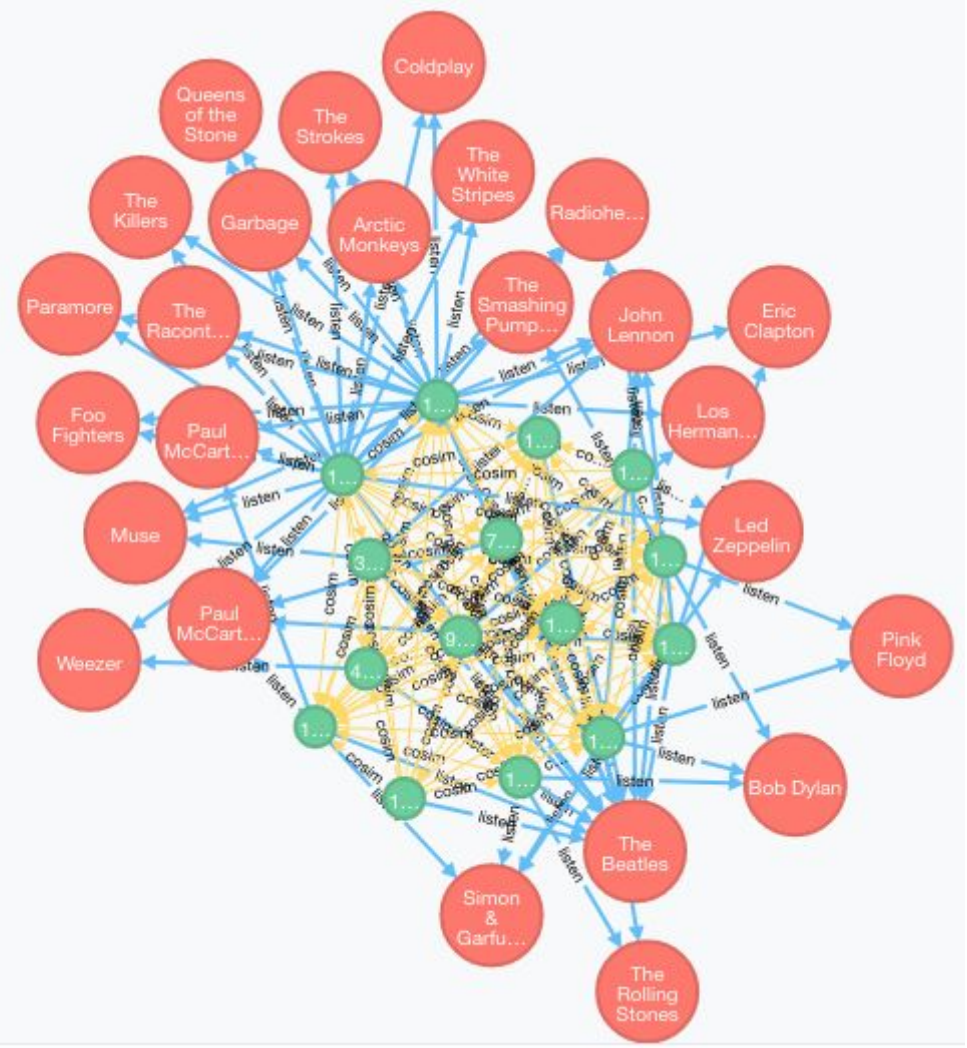
Nel seguente grafico possiamo notare che la comunità è di genere Pop ma abbiamo elementi anche Pop Rock come “Avril Lavigne” e “Evanescence” più Rock.



Query Cypher utilizzata per ottenere il grafico:

```
MATCH p=(u1:User)-[r1:cosim]-(u2:User)
MATCH q=(u1:User)-[r2:listen]->(a:Artist)
MATCH f=(u2:User)-[r3:listen]->(a:Artist)
WHERE "4345" IN u1.ks AND "4345" IN u2.ks
AND toInt(r2.weight) >= 1000 AND toInt(r3.weight) >= 1000
return p, q, f
```

Ancora un altro esempio di comunità individuata con algoritmo K-plex che dimostra una preferenza nel genere Rock:



Anche in questo caso possiamo notare un “rilassamento” del genere della comunità in quanto sono presenti maggiori sfumature, per esempio generi Pop Rock come “Coldplay” e “Paramore”, artisti non presenti nella comunità del genere Rock individuata da clique. Inoltre possiamo notare che sono comparsi nella stessa comunità gli artisti “The Rolling Stones” e “The Beatles”, fatto rilevante in quanto è noto che la maggior parte degli utenti che ascoltano uno dei due solitamente non ascolta l’altro, ciò non accade in clique.

Query Cypher utilizzata per ottenere il grafico k-plex descritto sopra:

```
MATCH p=(u1:User)-[r1:cosim]-(u2:User)
MATCH q=(u1:User)-[r2:listen]->(a:Artist)
MATCH f=(u2:User)-[r3:listen]->(a:Artist)
WHERE "5142" IN u1.ks AND "5142" IN u2.ks
AND toInt(r2.weight) >= 500 AND toInt(r3.weight) >= 500
return p, q, f
```

Louvain

Per validare l'algoritmo di Louvain, anche in questo caso abbiamo selezionato delle comunità su Gephi e visualizzandole su Neo4j. Louvain differentemente da Clique e K-plex, risulta avere un comportamente molto particolare in quanto non individua una singola comunità per genere musicale, ma più sub-comunità di diversi generi e gli artisti che fanno da ponte su tali comunità, ovvero gli artisti ascoltati da entrambi i generi.

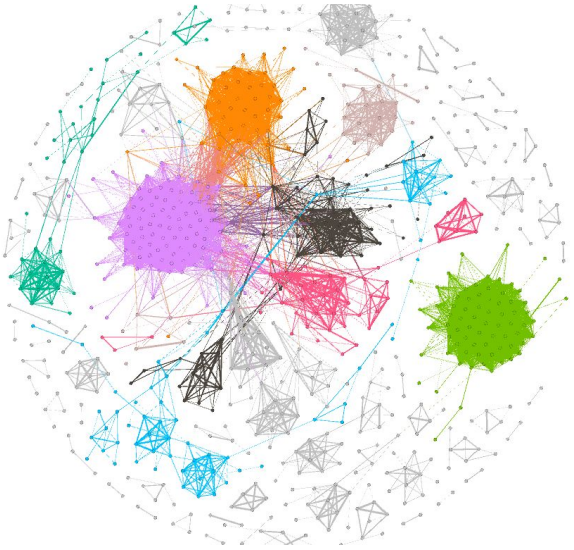
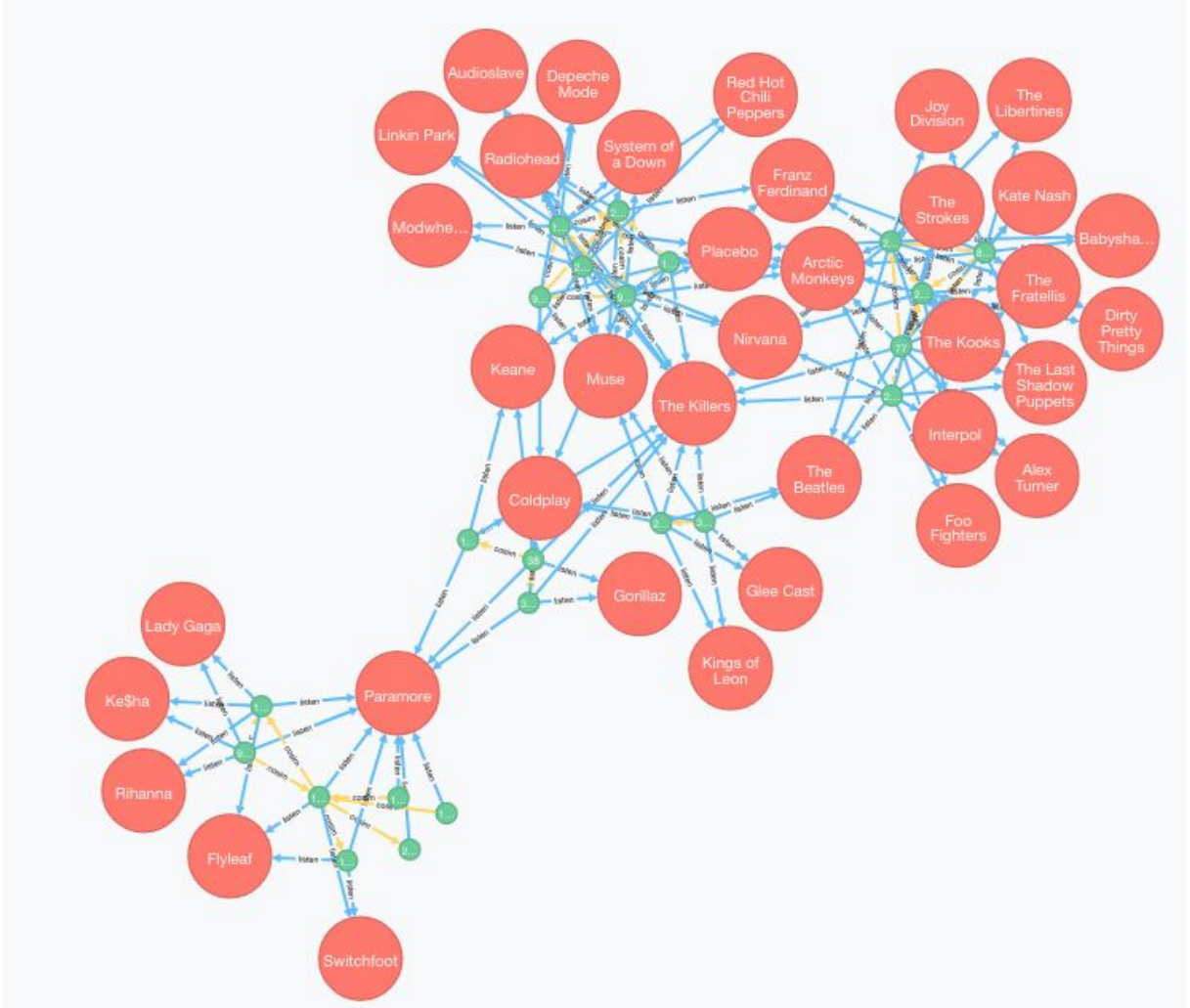


Grafico Gephi di tutte le comunità ottenute con Louvain nell'immagine a destra.

In questo grafo della comunità con ID 3 di louvain, è possibile apprezzare quanto descritto sopra:



E’ possibile notare come le varie comunità per genere musicale siano distribuite nel grafo:

- Genere Pop è situato in basso a sinistra
- Genere Rock prevalentemente centro-alto con diverse sfumature apprezzabili:
    - In alto a sinistra abbiamo prevalentemente un genere più tendente al Metal;
    - In alto a destra sfumature tra Indie e Alternative Rock;
    - Nel centro basso abbiamo prevalentemente Pop Rock.

Inoltre possiamo notare come alcuni artisti vadano a fare da ponte tra comunità di generi diversi o sottogeneri, come per esempio:

- Il gruppo Paramore con i suoi generi [Emo-pop, Pop punk, Pop rock] e “Coldplay” con i suoi generi [Alternative rock, Britpop, Pop rock, Pop] sono band a cavallo tra la comunità Pop da quella Rock.
    - Interessante notare come la band Paramore nonostante sia prevalentemente rock sia ascoltata maggiormente da una comunità Pop.
- La band “The Beatles” con i suoi generi [Rock, Beat, Pop] sono la band che separa la comunità Alternative Rock da quella Pop Rock.
- In fine la band “The Killers” con i suoi generi [Alternative rock, Post-punk revival, New wave, Indie rock, Rock elettronico, Dance rock] accomuna tutta la comunità del genere Rock.

Query in linguaggio Cypher utilizzata per ottenere il grafo:

```
MATCH p=(u1:User)-[r1:cosim]-(u2:User)
MATCH q=(u1:User)-[r2:listen]->(a:Artist)
MATCH f=(u2:User)-[r3:listen]->(a:Artist)
WHERE 3 IN u1.louvains AND 3 IN u2.louvains
AND toFloat(r1.weight) >=0.8
AND toInt(r2.weight) >= 400 AND toInt(r3.weight) >= 400
return p, q, f
```

Combinazione degli algoritmi di community detection

Oltre ad individuare e ad analizzare le singole comunità ottenute da ciascun algoritmo di community detection (clique, k-plex, louvain), sono stati combinati i risultati dei tre algoritmi su ciascuna coppia di nodi **i** e **j** del grafo, ottenendo una matrice NxN (N numero di nodi del grafo) contenente per ciascun elemento una tripla **[h,p,q]** che identifica:

- **h**: numero di **clique** a cui appartiene la coppia i e j;
- **p**: numero di **k-plex** a cui appartiene la coppia i e j;
- **q**: appartenenza o meno della coppia i e j a una comunità **louvain**.

Possiamo individuare almeno 8 casistiche per ciascun elemento della matrice, che per semplicità descriviamo in binario, indicando con 1 il fatto che la coppia di nodi i e j partecipa almeno ad una comunità dell'algoritmo k-esimo, 0 altrimenti. Da notare che la non presenza di un arco tra le coppie i e j non esclude il fatto di appartenenza alla stessa comunità, questo per definizione di k-plex e louvain.

Casistiche elemento [h,p,q] della matrice NxN

	h	p	q	
1.	0	0	0	Utenti **i,j** non partecipano insieme a comunità.
2.	0	0	1	Utenti **i,j** partecipano insieme a comunità louvain.
3.	0	1	0	Utenti **i,j** partecipano insieme a comunità k-plex.
4.	0	1	1	Utenti **i,j** partecipano insieme a comunità k-plex e louvain.
5.	1	0	0	Utenti **i,j** partecipano insieme a comunità clique. Identifica errore nell'individuazione della comunità da parte di k-plex o clique.
6.	1	0	1	Utenti **i,j** partecipano insieme a comunità clique e louvain. Identifica errore nell'individuazione della comunità da parte di k-plex o clique.
7.	1	1	0	Utenti **i,j** partecipano a comunità clique e k-plex. Occorre prestare attenzione alle cardinalità.
8.	1	1	1	Utenti **i,j** partecipano a comunità clique, k-plex e louvain. Occorre prestare attenzione alle cardinalità di clique e k-plex.

Ricordandoci che le relazioni tra utenti sono calcolate sulla base della coseno similarità sugli ascolti di artisti, ciò che possiamo analizzare di rilevante dalle casistiche descritte sopra, è il seguente:

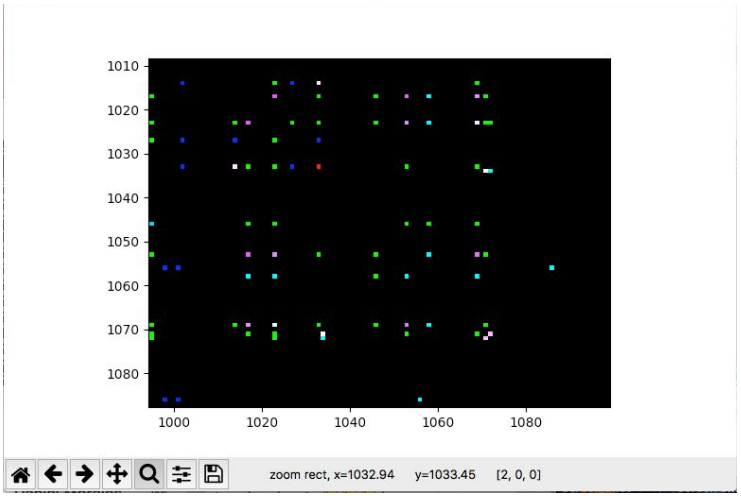
1. Nelle casistiche 1. e 8. abbiamo due comportamenti "forti", ovvero
    - nel 1. le coppie non sono mai nella stessa comunità.
    - nel 8. le coppie sono sempre in almeno una comunità in comune. Probabilmente nessuno può dire che questi due utenti non facciano parte della stessa comunità, a meno che gli algoritmi non siano sbagliati. Inoltre

occorre prestare attenzione alle cardinalità, descritte nel punto 3.

2. Nelle casistiche 5. e 6. individuiamo un possibile errore da parte dell'algoritmo di clique o k-plex in quanto se clique identifica la coppia i e j in una comunità questa deve essere individuata anche da k-plex.

3. Nelle casistiche 7. e 8. , ragionando non più in binario ma sulla cardinalità di h e p (clique e k-plex) deve essere verificato che h <= p, altrimenti finiamo nuovamente nel problema descritto nel punto 2.

Per consentire un'analisi visiva puntuale sulle coppie, la matrice è stata codificata in RGB.
Porzione della matrice in codifica RGB:



Codice dei colori:
- Rosso: solo/prevalentemente clique;
- Verde: solo/prevalentemente k-plex;
- Blu: solo/prevalentemente louvain;
- Celeste: prevalentemente k-plex e louvain;
- Rosa/Fuxia: tutti e tre;

La matrice consente di individuare un "quarto tipo" di algoritmo di community detection, generato dalla combinazione di clique, k-plex e louvain. Più nel dettaglio la matrice consente di descrivere l'appartenenza di ciascuna singola coppia alla stessa comunità o meno.

E' possibile trovare il codice relativo alla matrice di User all'interno del repository al seguente path: /community-detection-lastfm/code/**community-detection/community-core/matrix.py**

Conclusione

In conclusione i tre algoritmi (clique, k-plex, louvain) individuano comunità basate sul genere musicale ma con caratteristiche diverse.

Possiamo affermare che:

- Clique individua comunità prevalentemente di un singolo genere musicale con maggiore rigidezza rispetto gli altri due algoritmi.
- K-plex, individua generi con maggiore flessibilità consentendo anche la presenza di alcuni sottogeneri.
- Louvain, differentemente dai precedenti algoritmi, consente di individuare in una singola comunità più genere e sottogeneri. Inoltre, un fenomeno interessante è la presenza di artisti nelle comunità che hanno il ruolo di ponte tra generi e sottogeneri diversi ma correlati (overlapping communities).

Quindi in generale nel nostro dominio applicativo l'algoritmo Louvain Method, inizialmente sembra identificare male le comunità, invece è risultato in grado di congiungere comunità con generi diversi e con interessi per uno stesso artista piuttosto che dividerli.