Analyzing GEMSEC DEEZER Dataset using GraphX

1st Vansh Lal Tolani
DSAI
IIIT DHARWAD
22bds061@iiitdwd.ac.in

2nd Rajdeep Manik
DSAI
IIIT DHARWAD
22bds048@iiitdwd.ac.in

3th G Leeladitya
DSAI
IIIT DHARWAD
22bds024@iiitdwd.ac.in

4th Shanmukha D
DSAI
IIIT DHARWAD
22bds019@iiitdwd.ac.in

***Abstract*—This project focuses on the use of Apache Spark GraphX,which is one of the the librarires of Spark.Here we are going to take a dataset Gemsec Deezer which is a dataset taken from SNAP.Here we would be taking this dataset and applying 3 major algorithms of GraphX (PageRank,Connected Components,Triangle Counting,SVD++) and finding out valuable insights from our dataset.**

Terms-Apache Spark,GraphX,PageRank,Connected Components,SVD++

I. Dataset

In this study, the dataset utilized originates from Deezer, a music streaming service, collected in November 2017. The dataset represents friendship networks of users from three European countries: Romania, Croatia, and Hungary. Nodes in the network represent users, while edges denote mutual friendships between users. To maintain anonymity, nodes are reindexed starting from 0. The dataset includes CSV files containing edge information and JSON files containing users' genre preferences. Each JSON file associates user IDs with lists of genres they love. Genre notations are consistent across users, with each user potentially liking up to 84 distinct genres. The genre preferences were compiled based on liked song lists from the Deezer platform.

TABLE I
Gemsec Deezer Dataset information

Counrty	***Nodes***	***Edges***
Romania	41773	125826
Croatia	54573	498202
Hungary	47538	222887

A. Dataset Description

The Deezer dataset, consisting of friendship networks and genre preferences of users from three European countries (Romania, Croatia, and Hungary), primarily showcases interests in two main aspects Social conectivity and Music Genre Preferences.Overall, the dataset reflects the intersection of social networking and music streaming, emphasizing the importance of both social connectivity and music preferences in understanding user behavior and enhancing the user experience on Deezer.

B. Research Context

This dataset holds relevance for researchers investigating social network dynamics and user interactions within online platforms. It offers insights into friendship formation patterns and genre preferences among users of music streaming services, contributing to a better understanding of online social behavior.

C. Methodology

To facilitate analysis, the dataset has been preprocessed and formatted into an edge list. The preprocessing methodology involves the following steps:

- Creation of Local Spark Stand-alone Cluster: A local Spark stand-alone cluster is established to efficiently handle the large-scale data processing requirements.
- Reading of Edge List Using GraphLoader API: The dataset is read into the Spark environment utilizing the GraphLoader API, enabling seamless integration of graph-based operations.
- Construction of Directed Graph from Edge List: The edge list is utilized to construct a directed graph, which serves as the foundational structure for subsequent analyses and insights generation.

D. Community Impact

This dataset serves as a valuable resource for researchers exploring topics related to social networks, user behavior analysis, and music recommendation systems. Its availability in the Dataset folder within the parent directory streamlines access for researchers interested in leveraging it for further investigations and experiments.

II. Algorithm

The following Algorithms have been used:

A. Page rank

PageRank is an algorithm developed by Larry Page and Sergey Brin, the founders of Google, originally designed to rank web pages based on their importance and relevance.PageRank analysis could inform platform engagement strategies by identifying users with high influence scores who could be targeted for engagement campaigns, promotions, or community-building activities. Engaging with influential users can help amplify the platform's reach and foster user loyalty.

1) Nodes representation: Nodes in the network represent users of the Deezer music streaming service from three European countries: Romania, Croatia, and Hungary. Each user is assigned a node, and the nodes are reindexed for anonymity.

2) Edges: Edges represent mutual friendships between users. If user A and user B are friends on Deezer, there will be a mutual edge between their corresponding nodes in the graph.

3) Genre Preferences: Each user's genre preferences are provided in JSON files. The keys in the JSON files represent user IDs, and the values are lists of genres that the users love. Genre notations are consistent across users, and each user could like up to 84 distinct genres.

4) PageRank Calculation: We can calculate the PageRank scores for each user based on the friendship network graph. Users who are connected to other influential users (e.g., users with many mutual friendships) will have higher PageRank scores.

5) Interpretation: Users with higher PageRank scores are considered more influential within the Deezer music streaming community. These users are likely to have more mutual friendships and could potentially have a greater impact on the music preferences and interactions within the network.

By applying the PageRank algorithm to the friendship network dataset from Deezer, we can identify users who play significant roles in connecting others and influencing music preferences within the Deezer community.

B. Connected Components

The Connected Components algorithm is a graph algorithm used to identify sets of nodes in a graph where each node can reach any other node in the same set, directly or indirectly, and no node in one set can reach a node in another set. The Connected Components algorithm, when applied to the Deezer dataset, would identify cohesive groups of users who are mutually connected through friendships within the Deezer community. Here's how the Connected Components algorithm can be applied to the Dreezer data:

1) Network Representation: Each node in the network represents a user of the Deezer music streaming service. Edges between nodes represent mutual friendships between users.

2) Algorithm Execution: Apply the Connected Components algorithm to the graph formed by the friendship network. The algorithm will identify sets of nodes (users) that form connected components, where any user can reach any other user in the same set through mutual friendships.

3) Identify Clusters of Users: Each connected component identified by the algorithm represents a cluster or group of users who are mutually connected through friendships. These clusters may indicate cohesive communities within the Deezer user base, where users share similar interests and engage in mutual interactions. By applying the Connected Components algorithm to the friendship network dataset, we can uncover the underlying structure of user connections within the Deezer community and identify cohesive groups of users with shared interests and interactions. Using the Connected Components algorithm helps reveal the interconnectedness of users within the Deezer friendship network and identifies cohesive communities that contribute to the social dynamics of the platform.

C. Triangle counting

To count triangles in the friendship network dataset from Deezer, you can use the following algorithm:

1) Triangle Counting Algorithm: Initialize a counter to keep track of the total number of triangles. For each node in the graph: For each pair of neighbors (u, v) of the current node: Check if there is an edge between node u and node v. If such an edge exists, increment the counter by 1. The final count represents the total number of triangles in the graph.

2) Explanation: This algorithm iterates through each node in the graph. For each node, it considers all pairs of neighbors. If there is an edge between the two neighbors, a triangle is found. The algorithm counts each triangle three times (once for each node in the triangle), so the total count needs to be divided by 6 to get the actual number of triangles.

3) Complexity: The time complexity of this algorithm depends on the number of nodes and edges in the graph. In the worst-case scenario, where every node is connected to every other node, the time complexity is O(V^3), where V is the number of nodes. we can use this algorithm to count the number of triangles in the friendship network dataset from Deezer, which can provide insights into the level of interconnectedness and clustering within the user community.

D. SVD++

SVD++ (Singular Value Decomposition++) is an extension of the Singular Value Decomposition (SVD) algorithm used for collaborative filtering in recommendation systems. It considers implicit feedback such as user interactions or ratings without explicit feedback. Here's how we applied the SVD++ algorithm to the Deezer dataset:

1) Data Preparation: Load the dataset containing user interactions, such as listening history or liked songs. Load the friendship network dataset to consider social connections between users. Load the genre preferences of users from the JSON files.

2) Matrix Representation: Represent the user-item interactions as a sparse matrix, where rows represent users and columns represent items (genres in this case). Fill in the matrix with implicit feedback data, such as the number of times a user listened to songs of a particular genre.

3) SVD++ Algorithm: Initialize user and item matrices (U and V) with random values. Initialize user biases (bu) and item biases (bi) with zeros. Initialize the global bias (mu) as the mean of all ratings. By applying the SVD++ algorithm to the Deezer dataset, you can provide personalized music recommendations to users based on their implicit interactions, social connections, and genre preferences.

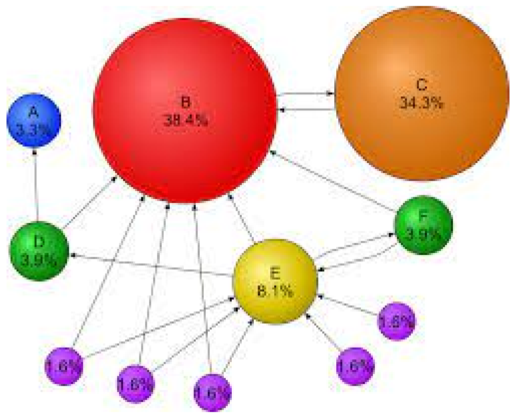


Fig. 1. page rank

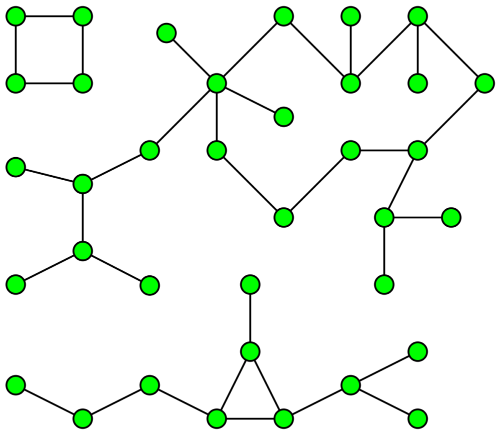


Fig. 2. connected components

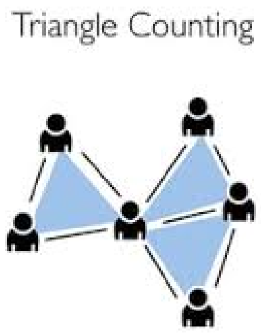
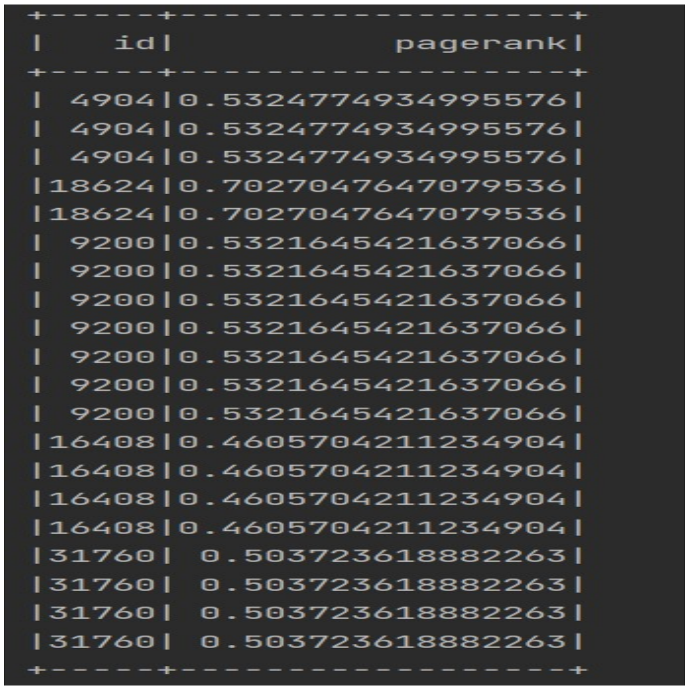


Fig. 3. Triangle counting

III. Result

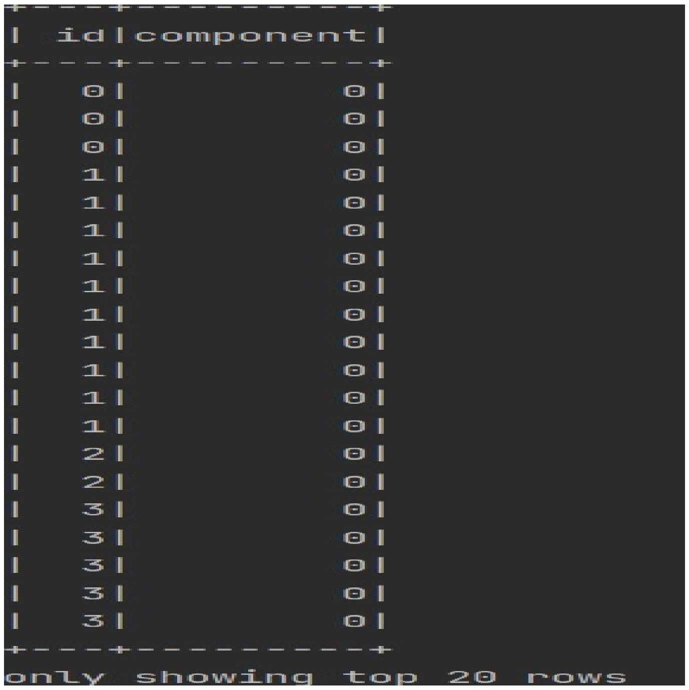
A. page rank



```
+-----+-------------------+
|   id|           pagerank|
+-----+-------------------+
| 4904|0.5324774934995576|
| 4904|0.5324774934995576|
| 4904|0.5324774934995576|
|18624|0.7027047647079536|
|18624|0.7027047647079536|
| 9200|0.5321645421637066|
| 9200|0.5321645421637066|
| 9200|0.5321645421637066|
| 9200|0.5321645421637066|
| 9200|0.5321645421637066|
| 9200|0.5321645421637066|
| 9200|0.5321645421637066|
|16408|0.4605704211234904|
|16408|0.4605704211234904|
|16408|0.4605704211234904|
|16408|0.4605704211234904|
|31760| 0.503723618882263|
|31760| 0.503723618882263|
|31760| 0.503723618882263|
|31760| 0.503723618882263|
+-----+-------------------+
```

Fig4: page rank result

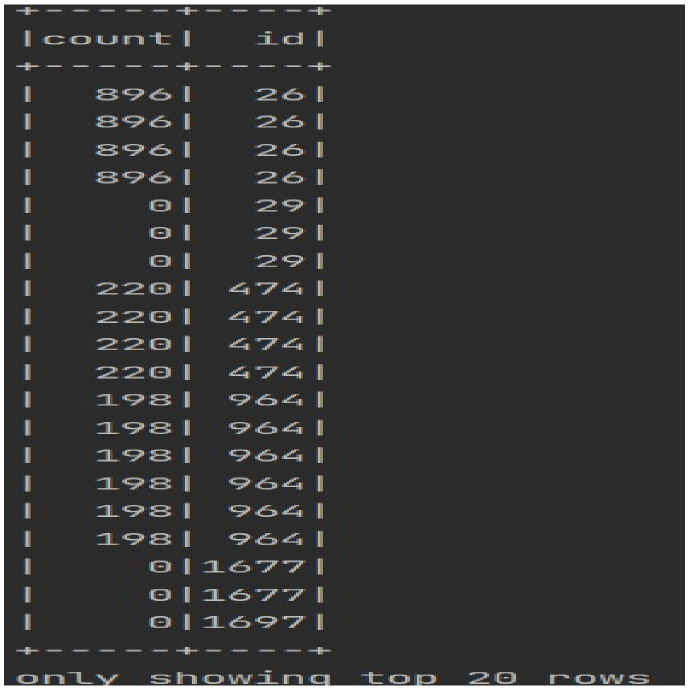
B. Connected components



```
+---+---------+
| id|component|
+---+---------+
|  0|        0|
|  0|        0|
|  0|        0|
|  1|        0|
|  1|        0|
|  1|        0|
|  1|        0|
|  1|        0|
|  1|        0|
|  1|        0|
|  1|        0|
|  1|        0|
|  1|        0|
|  2|        0|
|  2|        0|
|  3|        0|
|  3|        0|
|  3|        0|
|  3|        0|
|  3|        0|
+---+---------+
only showing top 20 rows
```

Fig5: connected components result

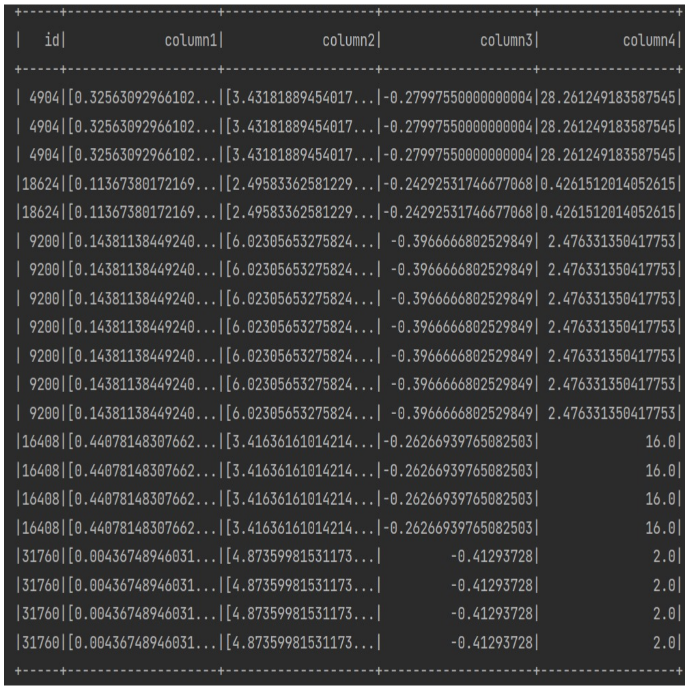
C. Triangle counting



```
+-----+----+
|count|  id|
+-----+----+
|  896|  26|
|  896|  26|
|  896|  26|
|  896|  26|
|    0|  29|
|    0|  29|
|    0|  29|
|  220| 474|
|  220| 474|
|  220| 474|
|  220| 474|
|  198| 964|
|  198| 964|
|  198| 964|
|  198| 964|
|  198| 964|
|  198| 964|
|    0|1677|
|    0|1677|
|    0|1697|
+-----+----+
only showing top 20 rows
```

Fig6: triangle counting result

D. SVD++



```
+-----+------------------+------------------+-------------------+-----------------+
|   id|           column1|           column2|            column3|          column4|
+-----+------------------+------------------+-------------------+-----------------+
| 4904|[0.32563092966102...|[3.43181889454017...|-0.27997550000000004|28.261249183587545|
| 4904|[0.32563092966102...|[3.43181889454017...|-0.27997550000000004|28.261249183587545|
| 4904|[0.32563092966102...|[3.43181889454017...|-0.27997550000000004|28.261249183587545|
|18624|[0.11367380172169...|[2.49583362581229...|-0.24292531746677068|0.4261512014052615|
|18624|[0.11367380172169...|[2.49583362581229...|-0.24292531746677068|0.4261512014052615|
| 9200|[0.14381138449240...|[6.02305653275824...| -0.3966666802529849| 2.476331350417753|
| 9200|[0.14381138449240...|[6.02305653275824...| -0.3966666802529849| 2.476331350417753|
| 9200|[0.14381138449240...|[6.02305653275824...| -0.3966666802529849| 2.476331350417753|
| 9200|[0.14381138449240...|[6.02305653275824...| -0.3966666802529849| 2.476331350417753|
| 9200|[0.14381138449240...|[6.02305653275824...| -0.3966666802529849| 2.476331350417753|
| 9200|[0.14381138449240...|[6.02305653275824...| -0.3966666802529849| 2.476331350417753|
| 9200|[0.14381138449240...|[6.02305653275824...| -0.3966666802529849| 2.476331350417753|
|16408|[0.44078148307662...|[3.41636161014214...|-0.26266939765082503|             16.0|
|16408|[0.44078148307662...|[3.41636161014214...|-0.26266939765082503|             16.0|
|16408|[0.44078148307662...|[3.41636161014214...|-0.26266939765082503|             16.0|
|16408|[0.44078148307662...|[3.41636161014214...|-0.26266939765082503|             16.0|
|31760|[0.00436748946031...|[4.87359981531173...|         -0.41293728|              2.0|
|31760|[0.00436748946031...|[4.87359981531173...|         -0.41293728|              2.0|
|31760|[0.00436748946031...|[4.87359981531173...|         -0.41293728|              2.0|
|31760|[0.00436748946031...|[4.87359981531173...|         -0.41293728|              2.0|
+-----+------------------+------------------+-------------------+-----------------+
```

Fig7: SVD++ result

ACKNOWLEDGMENT

I would like to extend our sincere thanks to our Assistant Professor Dr.Animesh Chaturvedi for providing us guidance throughout our project.This project has increased our technical knowledge

REFERENCES

[1] https://snap.stanford.edu/data/gemsec-Deezer.html
[2] https://spark.apache.org/docs/latest/graphx-programming-guide.html
[3] https://en.wikipedia.org/wiki/PageRank
[4] https://en.wikipedia.org/wiki/connected components
[5] https://github.com/Vansh-1007