Deezer Social Networks

Experimenting with dataset for finding different properties using algorithms

Lakshya Babel
Data Science and Artificial Intelligence
Indian Institute of Information Technology Dharwad
Dharwad, India
22BDS033@iiitdwd.ac.in

Suryansh Ayush
Data Science and Artificial Intelligence
Indian Institute of Information Technology Dharwad
Dharwad, India
22BDS057@iiitdwd.ac.in

Abhijit Singh
Data Science and Artificial Intelligence
Indian Institute of Information Technology Dharwad
Dharwad, India
22BDS054@iiitdwd.ac.in

***Abstract*— This report presents an analysis of friendship networks and genre preferences among users of the music streaming service Deezer in three European countries. That data used for finding different properties of dataset through different algorithms.**

Keywords*— *friendship networks, directed, connectivity, optimal, algorithms.

I. INTRODUCTION

The dataset acquired from Deezer in November 2017. This dataset comprises information regarding friendship networks and genre preferences of users from Romania, Croatia, and Hungary. The friendship networks are represented as graphs, where users are represented as nodes and mutual friendships are shown as edges. Furthermore, the genre preferences are compiled by considering liked songs list, and these preferences remain constant among all users. To protect user privacy, anonymization techniques have been implemented. The study conducted experiments using graph algorithms to analyze the Deezer dataset. The findings provided insights into network and included metrices of different properties.

II. NETWORK PROPERTIES

The table in *Fig. 1.1* provides information about the number of nodes and edges in each friendship network for the respective countries:

	Romania (RO)	Croatia (HR)	Hungary (HU)
Nodes	41,773	54,573	47,538
Edges	125,826	498,202	222,887

Fig 1.1: Network properties

III. DATA ANALYSIS

The data analysis done by Deezer can be useful for the solving of network clustering, the identification of influential users, and gaining an understanding friendly network as a whole.

Additionally, it also adds in analyzing the class of music preferences of individual Deezer users within the respective European countries. The analysis can be done by the use of various graph algorithms. There are numerous graph algorithm that can be used to conduct the study, Prim's, Kruskal's, Dijkstra's and Bellman-Ford algorithm. The minimum spanning tree (MST) and shortest path within a specific graph network can be found by these algorithms, which are useful for optimal and efficient data and user connectivity.

The table in Fig. 1.2 *(Demo 1)* provides the data related to the experiments done on graphs (G1, G2, G3) *as mentioned in question five of mid-sem*, meanwhile the table in Fig1.2 *(Demo 2)* provides information about the different four properties of the friendly network of Deezer, including average, median, minimum, maximum MST, as well as the shortest path. Here, edge weight is considered one because nothing related to edge weight is provided in the whole friendship graph network. For the shortest path in *Demo 1*, zero has been considered as a source node. The same study has been implemented on *Demo2 as well* for better efficiency and optimal findings.

Direct and most efficient connections between users belonging to which country, can be found by using minimum MST. Meanwhile, maximum MST helps in finding the most complex interconnections between users in the respective countries. The average MST helps in the study of connectivity patterns and the efficiency of patterns in three countries.

The minimum of shortest paths between interactions between users in respective countries helps identify the most related entities for music recommendation systems. The average shortest path is used to quantify the overall connectivity between users in three countries. A comparison of individual MSTs to the median one can be used as a performance evaluation metric for algorithms that generate the MSTs. The median of the shortest path helps to understand how easily users can find relevant content or connect with others. The maximum shortest path between users represents the highest connectivity between users in three countries.

IV. FINDINGS

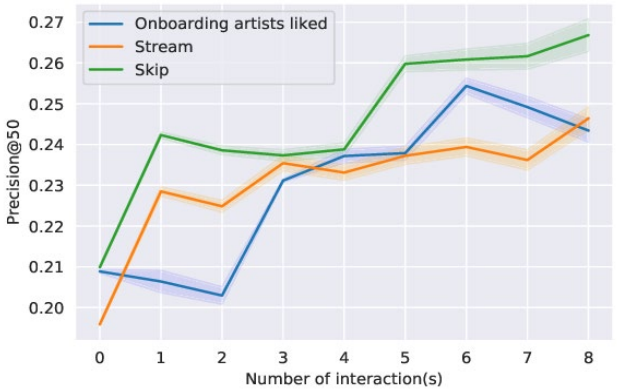

Fig1.2 the mean precision@50 scores of Deezer Semi-Pers. depending on the number of artists liked during the onboarding session, of streams, and of skips at registration day.

Properties / *Algorithms*	Average	Median	Minimum	Maximum
Prim's Algorithm	10.666	10.000	10.000	12.000
Kruskal's Algorithm	10.666	10.000	10.000	12.000
Dijkstra's Algorithm	27.333	30.000	22.000	30.000
Bellman-Ford Algorithm	27.333	30.000	22.000	30.000

(DEMO 1)

Properties / *Algorithms*	Average	Median	Minimum	Maximum
Prim's Algorithm	47960.333	47537.000	41772.000	54572.000
Kruskal's Algorithm	47960.333	47537.000	41772.000	54572.000
Dijkstra's Algorithm	66595.000	32987.000	3334.000	163464.000
Bellman-Ford Algorithm	66595.000	32987.000	3334.000	163464.000

(DEMO 2)

Fig 1.2: Application of Graph Algorithms

V. APPLICATIONS

I. Applications of (Minimum Spanning Tree) MST algorithms

1. Music Recommendation Systems: The tunes/artists are considered as nodes and their connections or relationships are considered as weighted edges (taken as 1). Applying MST algorithms i.e. Prims and Kruskal can help in differentiating different music recommendations for users.

2. Generation of playlist: Playlists which are connected, diverse, and representative of music preferences by users can be created by implementing MST algorithms to Deezer's graph.

3. Analysis of music genre: By using MST algorithms, one can identify the most important genres in the dataset, providing the perception of the structure and structuring of music genres.

II. Applications of Shortest path algorithm

1. The Deezer dataset may be used to determine the shortest path between two artists by considering the artists as vertices and the similarities between them based on different characteristics (for example, genre, cooperation, and popularity) as weighted edges. This can help visitors discover new musicians or explore various facets of an artist's career by pointing them in the direction of the most relevant or comparable artists.

2. The shortest path between two songs may be found by building a graph from the Deezer dataset with the songs as vertices and the associations between them (such as similarity and popularity) as weighted edges. This can be helpful in suggesting a run of tracks that progressively switch between various genres or styles, enabling users to discover a variety of music.

VI. BENEFITS AND LIMITATION

One of the major advantages of this work is that it helps in understanding the network thoroughly. However, it may not be able to compute the graph networks at a large scale because of network complexity and clustering.

VII. CONCLUSION

In conclusion, applications that deal in determining the shortest path between vertices in a network have a lot to gain from using the Deezer dataset. We can improve music recommendations, analyzes musicians/track owners and obtain insights into music cooperation networks by making use of this dataset. The collection offers a wealth of data from which graphs explaining connections between songs, artists, genres. By applying shortest path algorithms to these graphs, it is possible to resemble efficient and ideal paths, which provide improved music suggestions, effective event planning, and a deeper comprehension of the music business. Overall, by exploring shortest pathways, the Deezer dataset offers fascinating opportunities for enhancing numerous areas of music-related apps.

REFERENCES:

Network dataset: snap.stanford.edu/data/Deezer

Algorithms: algs4.cs.princeton.edu

ADDITIONAL RESOURCES:

www.kaggle.com

researchgate

Rozemberczki, Benedek & Davies, Ryan & Sarkar, Rik & Sutton, Charles. (2019). GEMSEC: Graph Embedding with Self Clustering

INPUTS AND OUTPUT OF CODES(LAB11)

LinkedList 1) Creating a LinkedList

Output	Input
	Roll: 4, Name: Suryansh, Age: 21 Roll: 3, Name: Madan, Age: 19 Roll: 2, Name: Abhijit, Age: 20 Roll: 1, Name: Lakshya, Age: 19
Students:	
Roll: 4, Name: Suryansh, Age: 21 Roll: 3, Name: Madan, Age: 19 Roll: 2, Name: Abhijit, Age: 20 Roll: 1, Name: Lakshya, Age: 19	
After deletion:	
Roll: 4, Name: Suryansh, Age: 21 Roll: 2, Name: Abhijit, Age: 20 Roll: 1, Name: Lakshya, Age: 19	

LinkedList 2A) Concatenate two given list into one big list.

OUTPUT	INPUT
Create List one:	
Number of elements in list1:	3
The value of node for list1: The value of node for list1: The value of node for list1:	1 2 3
Create List two:	
Number of elements in list2:	4
The value of node for list2: The value of node for list2: The value of node for list2: The value of node for list2:	4 5 6 7
Concatenated list: 1 2 3 4 5 6 7	

LinkedList 2B) Insert an element in a linked list in sorted order.

OUTPUT	INPUT
Enter the number of elements want to put in LL	6
Enter the value of Node Enter the value of Node Enter the value of Node Enter the value of Node Enter the value of Node Enter the value of Node	3 6 1 0 2 7
0 1 2 3 6 7	

LinkedList 2C) Always insert elements at one end, and delete elements from the other end

OUTPUT	INPUT
Inserted Element: Inserted Element: Inserted Element:	1 2 3
Deleted element: Deleted element: Deleted element:	1 2 3
Queue is empty	