IT TAKES TWO TO TANGO

Role Assignment & Modified Project Objectives
COMPSCI 2XB3 L01 Group 2

Ali Kariapper
Christina Mudarth
Hariesh Jay
Jack Buckley
Stefan Janovjak

15 February 2019

Selected Project

The projected we have selected is IT TAKES TWO TO TANGO. This is a project that allows users to search their friend network for people that like certain music genres. They can even sort their friends by how similar their music tastes are to theirs to find the perfect concert buddy or find the most popular music genres among friends who will attend a party, so they can play music that will please the maximum amount of people. All of this will be done by harnessing the preferred music genres and friendship information from a dataset from the music streaming service DEEZER in combination with specialized searching, sorting, and graphing algorithms—to help find friends of friends with similar music tastes— which will be validated by using unit tests.

Roles

Name	**Role**	**Responsibilities**
Ali Kariapper	Software designer	Ali will be responsible for designing the solutions to the tasks of the project. In particular, he will discover how to solve the big picture goal of the project by laying out specifications for creating different components of the project as well as how to piece them together.
Christina Mudarth	Algorithm research & Programmer	Christina will research algorithms to figure out what the most effective techniques for solving different tasks in the project are. She will also assist in programming the implementations.
Hariesh Jay	Tester	Hariesh will be responsible for testing the code so that it performs as expected on all inputs. In particular, he'll make sure the code is reliable and holds up to edge cases.
Jack Buckley	Project leader & Programmer	Jack will be responsible for making sure that all aspects of the project are coming along as expected by communicating with different team members. In particular, he'll make sure the project meets its milestones. Jack will also assist in programming the implementation of the project.
Stefan Janovjak	Log admin & Client interface	Stefan will be responsible for keeping the project log up to date to ensure everyone is on the same page with regard to the progress of the project. Stefan will also assist in implementing the client-side of the project, including implementing how the user interacts with the software.

1 Objective

This project serves to better connect individuals with their friends by discovering which music tastes they share in common. It will allow for one to determine which friend(s) one should ask to go to a certain concert and which genre of music to play at a party to please the most people by utilising information from the music service DEEZER.

2 Dataset & Project Input/Output

1. *Graph Embedding with Self Clustering: Deezer, February 13 2018*
   `https://snap.stanford.edu/data/gemsec-Deezer.html`
   This dataset holds all the DEEZER information for the project: both the preferred genres of music for each user (the nodes) as well as each user's friend group (the edges).

2. (a) Given a user's node ID and one of the 84 possible music genres, return a list of the user's friends who prefer this genre. This will be used if someone wants to quickly narrow down their friend group based on the criteria that their friend must like a certain genre of music.

   (b) Given a user's node ID and a list of some of the 84 possible music genres, return a list of the user's friends who prefer at least one of the given genres. If one wants to be less picky, they can still narrow down their friend list based on music genres; however, satisfying only one (or more) of the given genres will suffice.

   (c) Given a user's node ID, return the node ID of the friend with whom they have the most preferred music genres in common. This function can come in handy when one wants to find the person they should invite to a concert.

   (d) Given a user's node ID, return a list of genres that they and all of their friends share a common preference for. This is useful when one wants to create a soundtrack for a social event that all of their friends will attend.

   (e) Given a user's node ID, return a list of their friends ranked based on how similar these friends' music tastes are to that of the original user. This is good when one wants to invite multiple people to a concert.

   (f) Given a user's node ID and a list of music genres, return the closest acquaintance who shares all of these music genres in common. This will need to use a graphing algorithm to find the shortest path to this friend (or friend of a friend of a friend...), which will also ensure that the user gets to find the person with these tastes that they know most well.

   (g) Given a user's node ID and a list of the node IDs of their friends, return a list of genres that the user and the selected friends whose IDs are in the list share a common preference for. This will prove handy when one wants to create a soundtrack for a social event that only some of their friends will attend.

(h) Given a country's data file, return a list of the country's most preferred music genres ranked. This function can be used to find a country's overall music preferences which one can use if they need to play a soundtrack for a group of strangers.

(i) Given a user's node ID, return a tuple of the number of friends they have and a floating-point number representing the percentage of shared music genres (the number of shared genres in common / the total number of genres). This will be used to answer a question about if people who have more friends have more generalized music tastes.

3 What will it do? (Project Scope)

Using IT TAKES TWO TO TANGO is very easy. When the application starts, the client simply enters their ID (this simulates logging into DEEZER). Once they do this, they will see two options: "My friends are coming over. What music should I play?" and "I need a concert buddy. Search for a friend by music taste". Selecting the first option will bring up a window where the user can select who among their friends are coming over. They can do this by selecting each friend individually or selecting a checkbox that automatically selects all of their friends. Additionally, they have the option of searching up somebody manually by entering their user ID. This approach will make use of a searching algorithm to quickly locate this person's information. Once the friends who are coming over are selected, IT TAKES TWO TO TANGO will make use of a specialized sorting algorithm that finds the most common genres between the friends and returns the top three. This sorting algorithm is specialized in that it does not simply compare genre strings verbatim. Instead, this algorithm will make use of a lookup table to find each genres' category. For instance, under the "pop" category would be the categories "international pop" and "indie pop". Then, when the algorithm sorts by the most common genres, it will also take into account the overall category of the genre. This means that if two friends are coming over and both like " international pop", the algorithm will recommend "international pop" be played at the social event. However, if one friend likes "international pop" and the other likes "indie pop", and one genre is to be returned by the algorithm, the returned genre will be the overall "pop". We believe the best implementation of an algorithm like this would be to use a modification of a search engine algorithm. In this way, the algorithm will be able to make generalizations about individuals' music tastes by seeing that “contemporary R&B” is much more relevant to “R&B” than it is to something like “rock”.

The other option, “I need a concert buddy. Search for a friend by music taste”, will bring up a window where the user can enter one or more music genres. IT TAKES TWO TO TANGO will then return a list of friends who have these music tastes by using the aforementioned search and sorting algorithms. The user can select an option to make it necessary that a friend like all or only one of the genres given in the list. Additionally, if no friends have the given music preference(s), the user has an option to search for the first friend of a friend (of a friend, . . .) who does. In this respect, a shortest-path algorithm, such as Dijkstra’s algorithm, will be used so that the user is returned with the person that matches the criteria that they know most well.

4 Algorithmic Challenges

IT TAKES TWO TO TANGO will use several algorithms to achieve its goals. First, it will use a searching algorithm so that a user’s information and genres can be retrieved from the dataset. Since there are over 140 thousand users and 84 unique genres, the challenge here lies with finding values quickly. When it comes to sorting, the users’ friends will be sorted by how similar these friends’ preferred genres are to the original users’. A quick sorting algorithm, such as merge sort, will be necessary here. Moreover, another challenge will be placing weights on the genres. While there are 84 unique genres, some genres are subsets of other genres or are highly related. For instance, when sorting a person’s friends based on the general genre “rock”, a friend who does not have “rock” as a preferred genre but does have “indie rock” should be ranked ahead of a friend who only has “pop”. Lastly, IT TAKES TWO TO TANGO will use a graphing algorithm to find the first friend of a friend (of a friend, . . .) that likes a certain genre. The challenge is finding the shortest path from the original user so that the user is returned with someone they are most familiar with.

5 Project Plan

Have basic functions that return a user's genre info from the JSON file and their friend IDs from the CSV file finished.	Week 1
Have search function for specific genres finished.	Week 2
Have functions which return a user's friends who prefer certain genre(s) finished.	Week 2
Have function relating to top genres in country ranked finished.	Week 3
Have categorizing the 84 genres into related groups in a look-up table finished.	Week 5
Have sorting algorithm that takes into account genre similarity finished.	Week 5
Have function which finds top genres in common between friends finished.	Week 6
Have function which rank friends based on genre similarity to the original user finished.	Week 6
Have shortest-path algorithm finished.	Week 8
Have function which relates number of friends and genre similarity among these friends finished.	Week 8