Analysis

Introduction

My project aims to find a solution to being unable to find songs that are similar to those that a user already listens to. My program is aimed at a user base of teenagers who are attempting to broaden their music tastes, they may have recently found a new song they like but don’t know how to find more like it and would like to add some new songs to their playlists.

Problem Identification

*“3.1.1 Problem identification*

*(a) Describe and justify the features that make the problem solvable by computational methods.”*

My user base may currently get recommendations from friends and family members who have a wider range of music knowledge. As such the music that they get recommended does not stem from a full music database and will not be as accurate as a fully functioning computational solution with a wider access to comparison material that others may never have heard of.

In order to make this solution compete with just asking some friends to share preferences it should be able to match, if not be better, the ability to recognise songs that are similar of a human.

To do this I can take a variety of approaches which are currently used in existing systems. I can either design it to base its recommendations off songs that users input as liking both of, effectively building a map of songs. The alternative is to use a numerical break down of the music to make numerical comparisons.

Features that allow the solution to be executed computationally:

* Music can be broken down into numerical features as it is a waveform, there are many systems that have already done the breakdown of songs such as Spotify why makes all of the data public via their developer API.
* As all songs have numerical values associated with them using an algorithm to make comparisons and find trends between different songs.
* Searching a database of music to find songs allows a wider range of musical ‘knowledge’ and minimises the possibility of running into problems of not being able to recommend a new song due to not knowing the original.
* Storing the users preferences and the results given in a linked database means that there is a permanent record of songs they can come back to in future if they forget.

Being able to recommend songs in a manner that is close to mimicking a human recognising similarities cannot be done through conventional algorithms, a solution to this is using a neural network based algorithm which is defined as:

*“a computer system modelled on the human brain and nervous system”*

By modelling the system on the human brain and training the system from a user’s inputs you can get closer to how a human would decide if a song was close to the original inputs getting closer to mimicking a human decision making process.

Stakeholders of the Project

*“3.1.2 Stakeholders*

1. *Identify and describe those who will have an interest in the solution explaining how the solution is appropriate to their needs”*

The main stakeholders in this project are myself and others who want to widen their music listening base and discover more of a specific type of song. Currently the most popular way of listening to music is via Spotify and as such many are limited to the ability of finding new songs entirely based on what they are suggested. If there was a simpler way of finding songs that they might like based on only a selection of songs rather than using their entire listening base. Due to the pre-existence of Spotify my solution needs to be able to be easy to use and access as I am not aiming to compete with the music streaming aspect, the user should not feel it is too much hassle to open and search while listening to music.

I stand to gain a lot of experience in creating the project that I can then apply to more advanced projects at later dates, as well as along with my main stakeholders, gaining a program that can help create a wider music base.

Research

*“3.1.3 Research the problem*

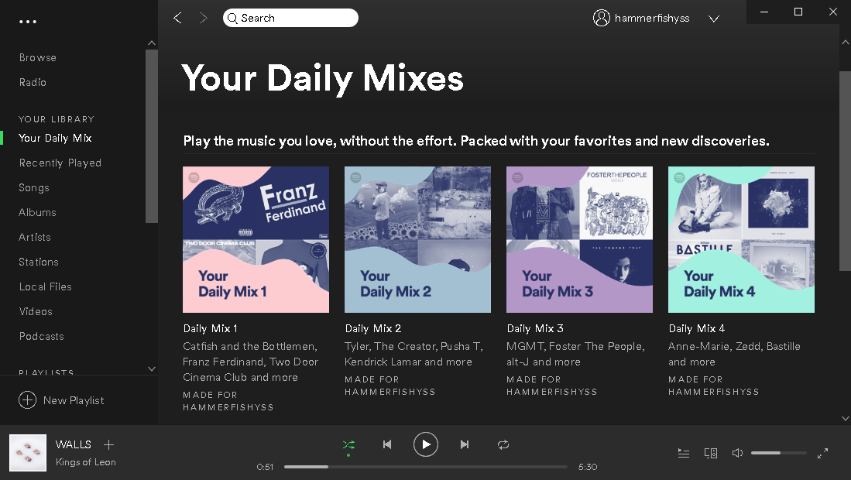
*(a) Research the problem and solutions to similar problems to identify and justify suitable approaches to a solution.”*

My project is not the first of its nature and as such many systems that provide similar experiences are available. Many music streaming services make use of algorithms to help listeners broaden their listening tastes. By analysing the ways that different programs present their data and how they reach their recommendations I will be able to make the best of previous designs and make a very accessible and flexible program.

1 – Spotify:

One of the major programs that makes use of music recommendation algorithms. As a very popular music streaming platform it attempts to broaden user’s music listening activity. One of their features is an automatic playlist generation that will match the user’s genre tastes, the “Daily Mix” playlists are auto generated, and such are powered by a powerful algorithm that can find songs that may not be related by artist but have a similar sound. They also provide a “Discover Weekly” playlist which as the name suggests is generated weekly. These suggest songs by artists that the user hasn’t listened to before, based on artists that they do listen to regularly.

Platform:

Spotify has been developed for a large range of different platforms due to its large user base. It has applications on the Google Play Store, Apple App Store and Microsoft Windows Store, on top of this it has a web player and applications for Windows, Mac OS, Linux and Chromebooks.

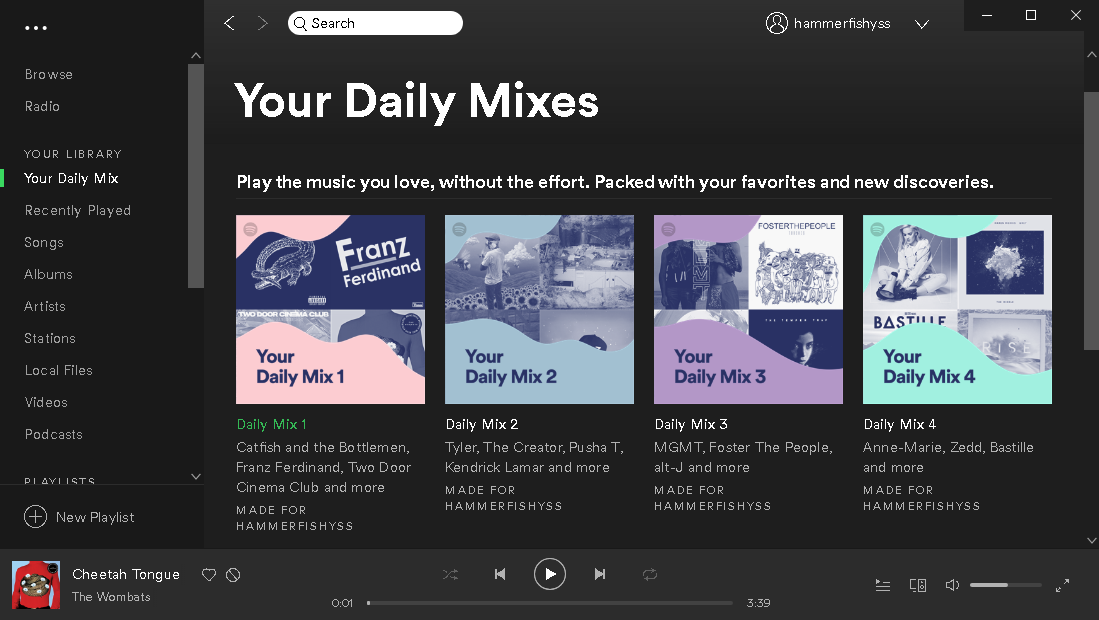
Price and Versions:

All these applications are free to download and use, but Spotify offer a Premium subscription-based service as well which costs £9.99/month. On top of the basic streaming functionality Premium allows users to listen un-interrupted by advertisements and skip an unlimited number of songs. The basic version of Spotify allows users to stream their music at 96kbps on mobile devices and 160kbps on desktop programs, this is upgraded by the premium version allowing users to stream at 320kbps.

These versions do not impact the recommendations the user gets, both versions have access to the personalised playlists.

Interface:

I will be primarily looking at the interface of the windows desktop application as that will be the platform my project will be developed for.

The user doesn’t get any choice in the songs that their recommendations are based on other than a like/dislike system by the song name when they are listening to a ‘daily mix’ or ‘Discover Weekly’ playlist.

The recommendations are presented in sets of playlists that are accessed by a side bar menu, this keeps the recommendation aspect out of the way of the main use (streaming music) but allows users who want to branch out an easy to use and dependable system.

Methods:

Spotify have a very advanced method of recommendation that considers multiple different ways of matching songs and then converge these into a verdict on whether the user will like them.

One of the main ways of recommending artists is based on a massive array of users and songs with Boolean values for each based on a like/dislike system for songs. This allows cross comparison of huge numbers of users to detect trends in shared taste.

The other method uses a neural network to decompose the auditory features of songs allowing songs to be recommended based on the similarity of the sound. This is a lot more advanced than simple trend comparison, allowing for different songs to be recommended outside of songs by artists that are linked to current favourites.

Sources:

<https://medium.com/s/story/spotifys-discover-weekly-how-machine-learning-finds-your-new-music-19a41ab76efe>

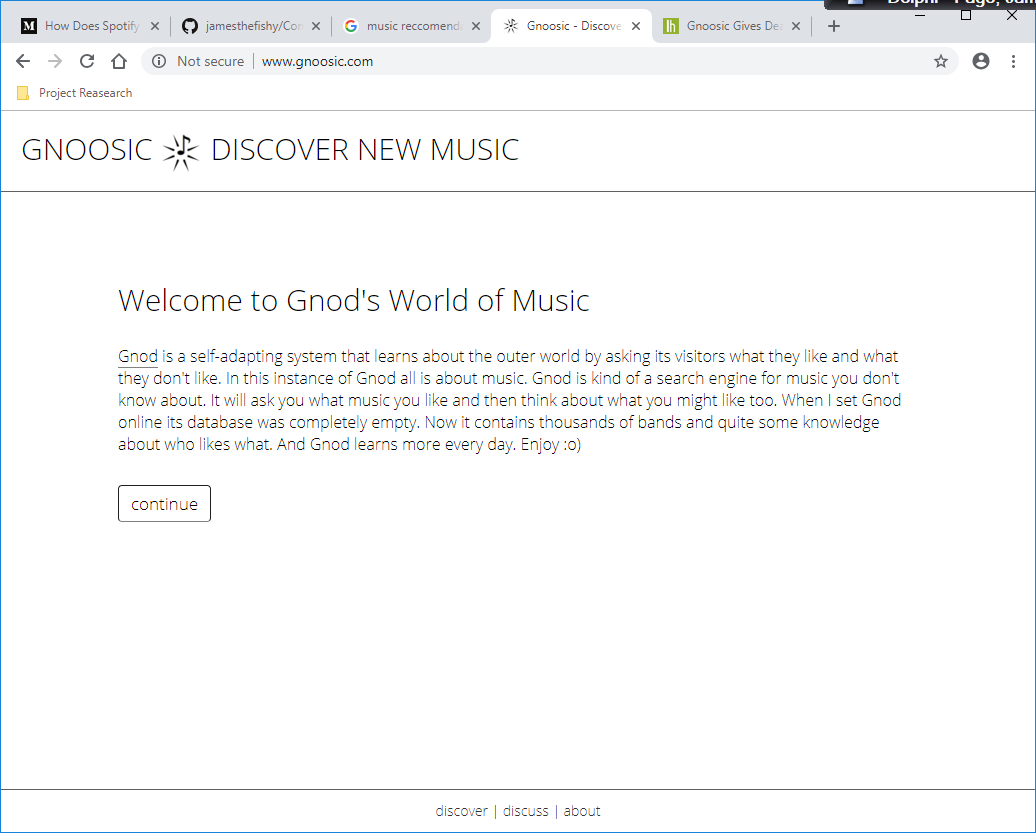
<https://support.spotify.com/is/using_spotify/the_basics/what-is-spotify/>

2 – Gnoosic:

Gnoosic is web-based algorithm that takes in 3 artists from the user and recommends a different artist which the user can either select 'like', 'dislike' or 'Don't know'.

In comparison to Spotify it does not have as many features due to it being a simple recommendation site and not having the streaming functionality.

Platform:

Gnoosic is part of a collection of recommendation algorithms called The Global Network of Discovery, all of these services are web-based so available on all web connected devices.

Price and Versions:

As it’s a web-based program, and anyone can access it online, it is completely free for anyone to use.

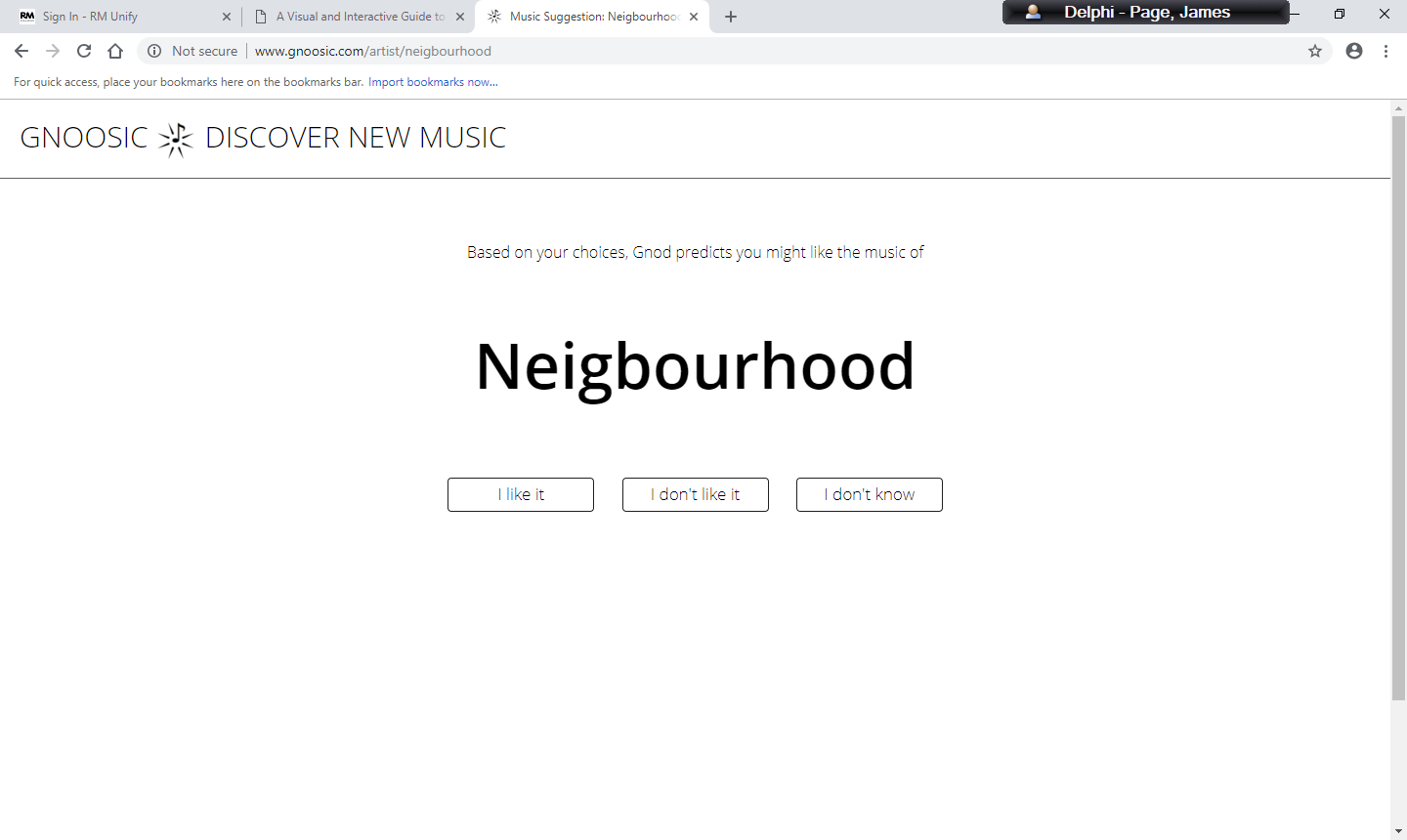
It only has one version but as part of the Gnod group of programs there is a connected website called music-map which shows how the recommendations are linked.

Interface:

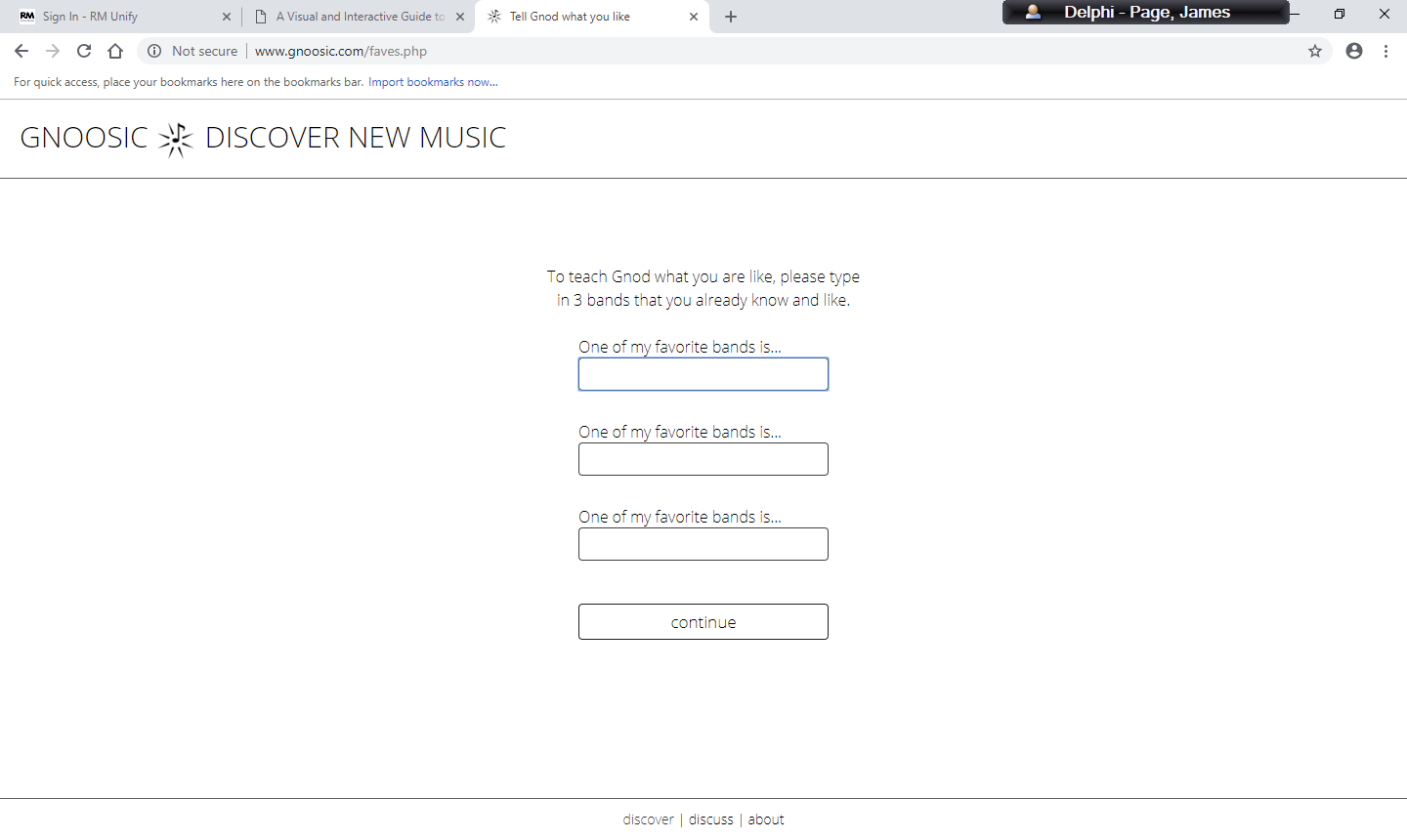
Gnoosic uses a simple white form user interface that allows the user to enter 3 artist choices and then shows you the name of a different artist that is linked to a the 3 entered by the user.

The connected site ‘music-map.com’ allows you to see the artists that are deemed ‘nearby’ to the selected artist in with the closer they are on the visual ‘map’ the more likely you are to like them.

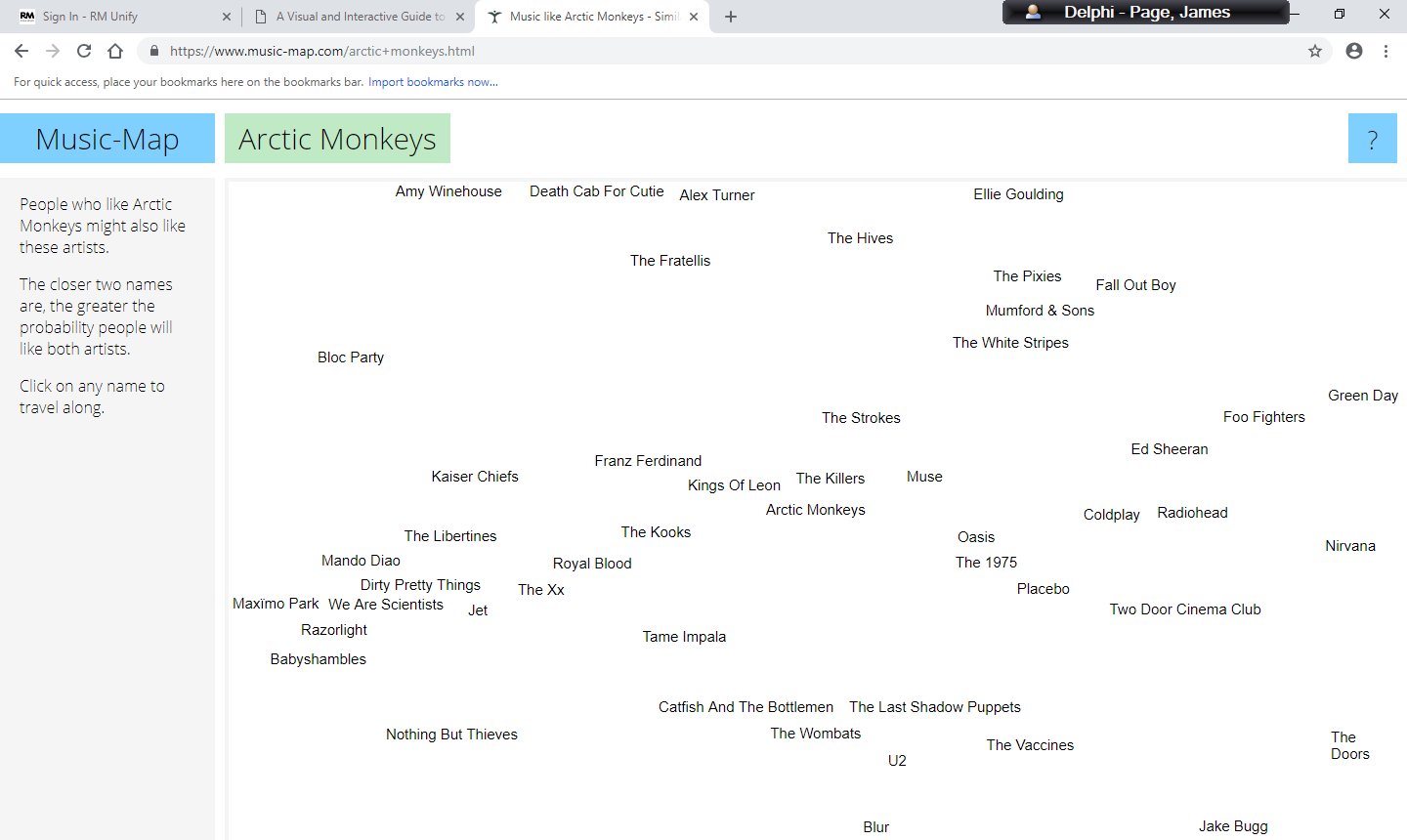
There is also an account feature which allows the user to save their preferences and find more artists in future.



The Results page



The input form



The connected music-map.com site that shows artist who are viewed as close to the entered artist stylistically.

Methods:

This webpage makes use of similar techniques as the first point mention in the Spotify section. By asking users to enter 3 artists they enjoy the site adds links between those artists entered allowing the algorithm to ‘learn’ about music tastes:

*‘Gnod is a self-adapting system that learns about the outer world by asking its visitors what they like and what they don't like’*

This means that as well as finding songs when they enter a search they are directly impacting how results may come out next time.

Sources: <http://www.gnoosic.com/about>

*https://www.music-map.com*

Summary:

The two different programs I looked at take very different approaches to recommending users new music mainly due to the differences in the environment that the recommendation is applied.

As mentioned, Spotify being primarily a streaming platform means it has a different range of features that a simple recommendation program will not require. However, the ability to create playlists of the songs recommended to you is a very useful feature and the recommendation algorithm is a lot stronger and more like what I hope to replicate as it does not require a large amount of users to build an accurate recommendation.

The benefit of the Gnoosic system is that although the recommendations themselves maybe not as strong as the Spotify system, the simple interface allows you to find new songs based on an exact song rather than just your overall listening preferences. This alongside the account system saving your liked artists makes for simple and quick use, that is just as functional for a new user as for a long time user.

Features Expected:

Based on my research, to be able to compete with current solutions I should design my project to meet the following criteria:

* A simple and understandable user interface that allows the user to enter 2-4 songs they enjoy and would like recommendations based on.
* Songs recommended based on a neural network based algorithm using Spotify developer API to retrieve song data to use as input values.
* Train the neural network to find songs for each user saving the trained configuration to a database so the user can use it again.
* Save the songs found to return positive results when passed through the algorithm to a database so the finalised results can be recalled as a ‘playlist’.
* Recallable neural network setting via a username system.

Limitations:

One of the main benefits of Spotify’s solution is that due to the platforms pre-existing music streaming services the recommendations can be easily and quickly converted into playable playlists allowing users to get an audible result immediately rather than just a song name which is all my system would be able to achieve.

On top of being unable to make use of the integration of a streaming system, an advantage of the large user base is being able to look a pre-existing trends, something that I would not be able to design my program to do at launch as a large set of data collected from a multitude of users would be required to get even the most simple of recommendations.

Proposed Solution

*“3.1.4 Specify the proposed solution*

1. *Specify and justify the solution requirements including hardware and software configuration (if appropriate).“*

My project can be split up into a few core sections/requirements which I will need to complete in order to fulfil all of the features and functionality expected.

A comprehensive main/home screen:

The program needs a main screen for the functionality to rotate around, from which they can access all functionality.

Input system:

In order to get their recommendations the user needs to input their original songs.

Results Screen:

After the inputs have been processed and songs have been found that are similar, the user needs to be able to be shown them in an easy to read manner.

Spotify API requests:

In order to compare and recommend songs the program requires data of each of the entered songs and the songs it will compare against, to do this it needs to make requests with the Spotify API.

Neural Network:

The recommendations will be made via a neural network which takes input values and through a series of biases and layers give an output.

Neural Network Training algorithm:

In order to make correct and accurate outputs the neural network will need to be trained for each user based on the songs they inputted.

Inputs:

The system has one main input system from the user, but by taking this user input the system can effectively find more input data from the external API I plan to take data from.

|  |  |
| --- | --- |
| Input | Details/What is it used for? |
| Song names | The user should input a set of songs they wish their recommendation to be based off. |
| Song data | From the song names, each of the songs has a set of numerical values associated with it, these values are used as inputs for the neural network algorithm. |
| Neural network inputs | As well as the original input to the neural network, each layer has a set of outputs that are used for the next layer. |
| Username choice | When the neural network is trained the values should be able to be saved under a username/keyword. |
| User data recalling via username | When a user uses the program again, they should be able to enter their username chosen prior to reuse the neural network/add additional training. |

|  |  |
| --- | --- |
| Output | Details/How is the output reached? |
| Song data | After the user decides on their songs the API returns the data of each song. Could be hidden from user. |
| Neural network outputs. | In the neural network each layer has a set of outputs as well as the prior mentions inputs, as well as the final output which decides on the Boolean value of like/dislike. All of these values are hidden under the mask of the algorithm. |
| Found song list | The main output the user gets to see, a list of the songs found based on the original input. |
| Error Messages | When the user is able to make an input such as the song names or as a request to recall a username’s data if the values entered cannot be found then the user should be informed of the error. |

There is no need for any users to have more access to the program than others as it is an automated system that should not need any administration to maintain.

I will develop my program on Windows 10 using:

* Delphi RAD Studio XE8
* Microsoft Access

I will be developing at home and at school, therefore the hardware I will be using will be limiting on the machines with a lower hardware specification:

* Intel i5-6400 CPU Quad core 4 threads @ 2.70 GHz
* 8GB RAM

When developing at home I will have access to more powerful hardware,

* AMD Ryzen 5 1400 CPU Quad core 8 threads @ 3.40 GHz
* 8GB RAM
* GTX 1060 6GB video card 1280 Core @ 1569 MHz

While having access to the more powerful hardware at home will not necessarily improve the developing ease, it may be interesting to see how performance is impacted by having access to more powerful hardware. The more intensive part of the program will be the training algorithm which could potentially take a lot of time to run in some situations and having a faster clock speed and more processing threads may improve training times. The possibility of creating compatibility with additional cores on a video card could decrease the processing time massively.

I have no allocated budget to complete the project, as such all work has to be done by me and should be completed using software I own already as I cannot purchase any additional licence’s. The project must be completed by February Half term which introduces a time constraint to the project but it should be a long enough window to fully implement the my solution. There should not be a need for the user to have any prior experience with the system and should be intuitive to new users so showing all of the data being ran through the algorithm could be daunting and cause confusion.

*“(b) Identify and justify measurable success criteria for the proposed solution.”*

Success Criteria

1. The graphical user interface is not daunting and easy to understand and use, there is no need for instructions and the program is presented to be intuitive for all users no matter ability/prior experience.
2. The program can take a user’s input of song names and accurately find the songs and pull the data associated with the song from the Spotify API.
3. The system should be able to pass the values through a neural network algorithm and return a Boolean value, predicting like or dislike.
4. The program should be able to train neural network using the data fetched from the songs the user inputs
5. The program will have a method of ‘logging in’ by using a key word to restore settings of a trained network to allow multiple uses of the program without having to wait for the training processes each time.
6. The program should be usable on a variety of hardware without having to wait a long period of time to get recommendations on slower hardware.
7. The user is presented a clear list of songs after the process has been run which give details of each song allowing them to easily be found and then listened to.
8. The program is finished and able to be distributed by the end of February 2019.

1. The program should be robust and should not run into errors in the code, any potential errors should be caught and return a message to the user informing them of the problem with their inputs.
2. The program can be run from download without any changes made to set it up allowing anyone to use the application without any maintenance at a later date by a ‘administrator’ user, therefore all users have access to the same information.