Deep Content-Based Music Recommendation - Specification

Problem statement (gap in existing provision)

There are millions of songs in existence that are now available to us digitally. In today's top music streaming platforms (Spotify, Apple Music, Tidal etc.) we find ourselves being recommended a multitude of songs. In the last decade, music recommendation has become increasingly important as it can enrich a listener's experience on a streaming service. A popular technique used in music recommendation is **Collaborative filtering**.

Collaborative filtering determines user preferences **based on historical usage data**. For example, if two users listen to largely the same set of songs, their tastes are probably similar. This information can be used to make recommendations to a user based on what similar users are listening to.

However, Collaborative filtering is **content-agnostic.** They do not care about information carried within a song. This means Collaborative filtering can be applied to a variety of recommender systems e.g. movies and books.

This feature is also it's biggest flaw. As it solely relies on usage patterns, popular songs are much easier to recommend than new and unpopular ones which have less historical data. This means the recommendations can be boring and predictable more often than not. If there is no usage data on a song, the collaborative filtering approach breaks down.

This is known as the **cold-start** problem. New songs that have not been consumed before cannot be recommended. In addition to this, items that are only of interest to a niche audience are more difficult to recommend because usage data is scarce, creating a **popularity bias**. This problem is what I aim to tackle by using content-based recommendation, which takes musical content into consideration.

By recommending songs in this manner, I hope that my system will be able to recommend new and unpopular music. Listener's should be able to listen to wonderful artists that they've never heard of.

Objectives

- Main Goal: Build a content-based music recommendation system; given a set of 10 songs return a playlist of 20 similar songs
    - Sub Goal: Build a system that can categorise music into cluster's based on a similarity metric

- Stretch Goal: incorporate a convolutional neural network into the system

To achieve these goals, the following tasks must be achieved:

- Research existing content-based music recommendation approaches that use neural networks
    - Read academic papers on music information retrieval (MIR) which is the related field of the project which is concerned with extracting data from music that can be modelled

    - Read tech journals & blogs related to machine learning and neural computing which will allow me to gather more information about different ways to solve the problem
- Learn how to program in python
    - Read python tutorials related to machine learning
    - Enrol in online python courses available on platforms such as Codecademy
- Find a suitable dataset to train the model, that can supply audio samples of songs and it must be of a suitable size to provide a realistic setting
    - The dataset will feed training data to the machine learning algorithm so that it can have a base layer to compare songs with
    - The dataset will also contain test data which we can compare the actual result to the expected result
- Decide on a representation of the audio signals so that they can be easily compared
    - The raw song audio needs to be processed so that it can be mathematically modelled
- Develop a suitable similarity metric to measure similarity between audio signals
    - This system will incorporate the use of deep learning, to decide what features will be used to define similarity between songs.

Methods

As this project has an equal part of programming and research, I will decide methodologies for both aspects.

Software development methodology

I will use a scrum (agile) approach when developing the system because of the following reasons:

- I do not have any experience coding a recommender system therefore I need a flexible methodology because I am unable to fully gauge the time needed to develop the system.
- Using a scrum approach allows me to weekly reflect on my progress with the system with review meetings, so that I can give regular updates to my supervisor
- If I hit any obstacles, I can deal with them in a timely manner using sprint cycles

Code management

I have decided to use **GitHub** as my version control software. This will ensure that my work is always backed up in the cloud and I do not have to worry about losing data on my local machine. It will allow me to access previous versions of my work and rolling back if needed.

Evaluation methodology

I will be able to evaluate my project by reviewing my objectives at the end and seeing what goals were met/not met and why. This will evaluation will include reflection on the project itself and improvements that could be made in a future project.

Data analysis methodology

When analysing the accuracy of the system, data visualisation will be used to help verify this. This is because it will make the results easier to comprehend rather using just figures. It will allow me to recognise any patterns and relationships in the data.

Timetable

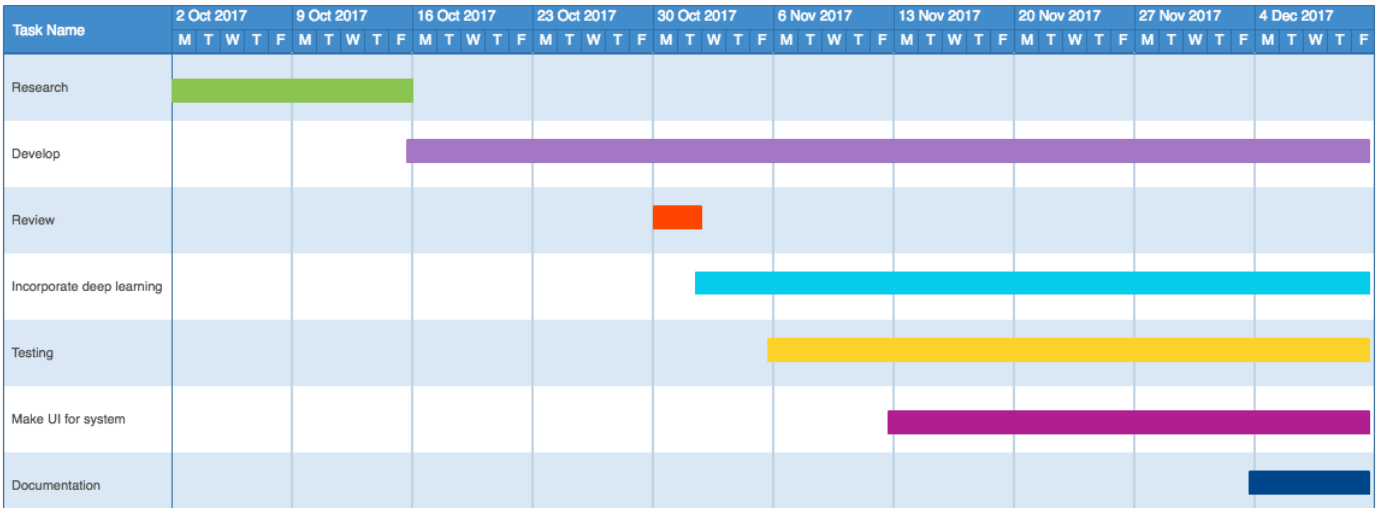

Figure 1: Term 1 – Project Timeline

In the first term, I have planned a heavy focus on the development side of the of the project. Development begins in week 3, this allows me to handle all possible contingencies. Some parts of development may take longer than expected because a majority of technologies I will be implementing are new to me. Therefore, starting development as early as possible allows me to countermeasure this.

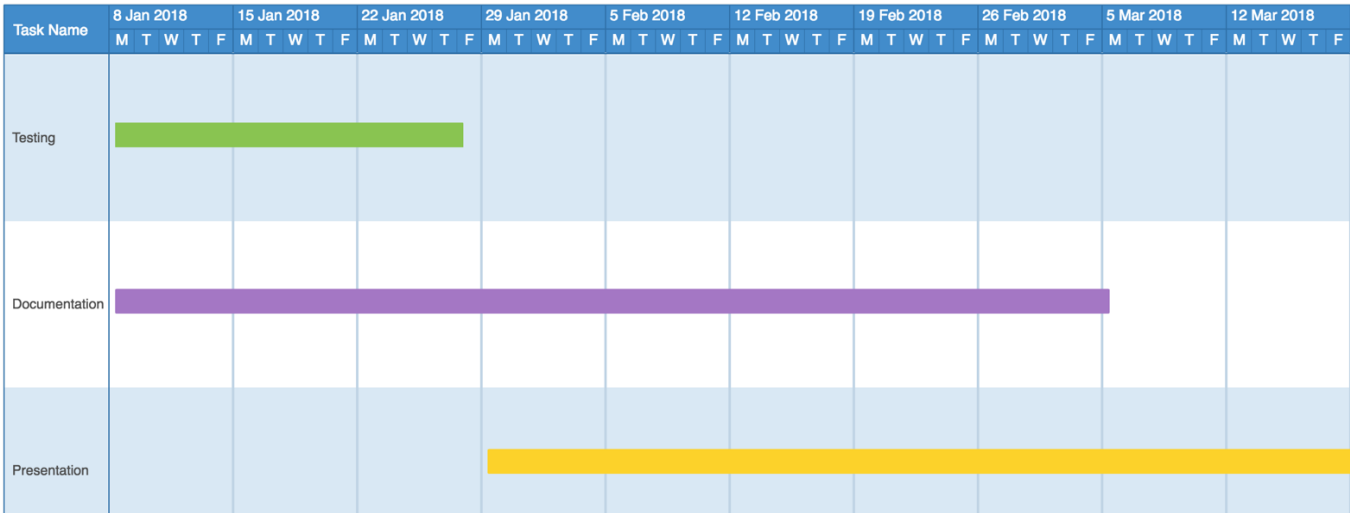

Figure 2: Term 2 - Project Timeline

In term 2, my time will be spent primarily on the documentation and the presentation. I will be continually refining the recommender system in my spare time and testing throughout the first half of the term.

Resources to be used

Knowledge resources

- Articles and research papers regarding different areas of the project. I will obtain them using search engines and platforms such as Google Scholar (a database of academic papers)

Hardware resources

- The project's hardware requirements will be satisfied by the use of my laptop which has enough processing power (Intel core i5 & 8gb ram) to handle the computations that will be carried out.

Software resources

- The project will be coded in **Python** as there is a wide selection of free libraries which I can use for machine learning, data transformation and data visualisation. As python is a

growingly popular language, the community is very responsive and helpful. This means there is a wealth of online resources to guide me throughout the development of the system

  - **TensorFlow** is a free SDK built by google will be used for the machine learning part of the project. It is an open source artificial intelligence library which uses data flow graphs to build models. It allows developers to create large-scale neural networks with many layers. TensorFlow is mainly used for: Classification, Discovering, Prediction and Creation.

  - **YAAFE** is a free python library will be used in to process the raw audio signals into MFCCs and allow me to extract features from the songs. It is a popular library in the field of music information retrieval (MIR) because it allows the programmer to easily specify the representation to convert the raw audio into and there is a lot of extra parameters that can be set also.

- The system will be coded within the open source text editor **Atom**, made by GitHub. I chose to use this specific editor because of it's customizability. There are thousands of packages available for free, that can greatly enhance your coding experience. It's integration with GitHub means I can easily commit changes within the UI.

Data resources

- I will use the million song dataset (MSD) as training data for the recommendation system. It is a freely-available collection of pre-computed audio features and metadata for a million contemporary popular music tracks. However the pre-computed features will not be extremely useful as I will need to generate the features in real-time to feed into the recommender
  - To obtain the actual raw audio I will have to use 7digitals API which will allow me to download 30 second samples of each song

Ethical considerations

All the data that I will be collecting will be provided by the publically available Million Song dataset therefore I will not need any ethical consent as the data is not sensitive.

All libraries and API's that will be used are licensed and legally approved.