Network Tour of DataScience:
XXX

A. Buchegger & L. Biotto & J. Tapparel & T. Tuuva

1 Introduction

It can be tough to know what we would like to listen when faced with an archive of thousand of musics. For our project we wondered how we could automatically generate a list of songs that would correspond to one's taste. We will explore the features of the Free Music Archive (FMA) to generate a graph that will contain many songs of selected genres. These songs can then be sorted and a playlist be generated out of a region of this graph. We will explore different methods of generating playlists. At the end, we will show how this tool can be used to generate a free alternative playlists from non-free musics.

2 Generating the graph

3 Including our own song

To include our own music into the graph we first need to extract its features, this is done by using the provided features.py script(TODO REF), there are 518 features in total. Then we can include the song in two ways:

- Recompute the graph adjacency matrix for every included songs
- Create a KD-Tree of the graph, then search the nearest point in the K-features space for the included songs

3.1 Inclusion by adjacency

TODO

3.2 Inclusion by KD-Tree

Instead of recomputing the adjacency matrix and the corresponding graph which has a high computation cost, we can generate another kind of graph directly from the features, in our case we chose a KD-Tree. An example of a 2D-Tree is presented in the figure (1)

This kind of structure is conceived to quickly find the nearest neighbour in an effecient way. We decided to consider that the closest neighbour to our newly included song in the feature space are the same node. Hence, when we want to extract a playlist in the other methods, we will use the representating node as the new song.

GLAAA

Figure 1: Example of a 2D-Tree

4 Generating the playlist

To generate a playlist we explored three distinct methods:

- Nearest neighbours in the K-Space of the 518 dimensions features spaces using KD-Tree query
- Euclidean distance minimisation with the 2D-Graphs coordinates
- Heat diffusion in the graph

4.1 518D features exploration

First, we can create a playlist directly from the euclidean distance between features. This property should result in similar features, such as rythm, or height of voice of example, but being more adventurous in regards to genres. One could for example end up listening to hip-hop when starting from rock.

4.2 2D graph exploration

Secondly, we select the closest songs around the selected music (node) in the graph. This effectively results in a kind of circle around a point. This type of playlist will thus be very targeted towards a specific kind of song. For this purpose we use the simple euclidean distance in our 2D graph.

4.3 Graph heat diffusion

Finally, we generate a heat diffusion around selected nodes, and create a playlist from the ”warmest” neighbours. This selection should result in a kind of path between different closely connected songs. When listening to a playlist, it is nice to be able to tell how adventurous we feel regarding the inclusion of other styles. We thus leave the choice to the user to select its own prefered playlist generation.

Figure 2