LINUS - MUSIC RECOMMENDATION SYSTEM 2016

Harpreet Singh
University of Victoria
hsingh91@uvic.ca

Ishmeet Singh Kohli
University of Victoria
ishmeet@uvic.ca

Harmeet Singh Saimbhi
University of Victoria
hsaimbhi@uvic.ca

ABSTRACT

There are a lot of activities in which the user wants instant music but doesnt want to spend time sitting and creating playlists. The goal of this project is to create a playlist generator that takes as seed either a short recording of audio or asks the user to define a preset (for eg. an activity like running, studying, programming, etc.) and then listen to a few songs. The user provides feedback to these songs which the system uses to create a custom playlist on spotify. While the playlist is running, the user can provide feedback by liking, disliking or skipping songs which will cause the playlist to be updated. To improve recommendations, in addition to using audio features we will also take into account social tags. [8]

1. INTRODUCTION

Due to the internet, we now have millions and millions of songs available to us at a click of a button, but ironically this has made it even harder to choose music to listen to. A number of companies have come up to solve this problem but with different approaches. For eg: Pandora tries to solve it by employing actual people who sit all day and classify songs, but it is a model that is not scalable. Last.fm uses colloborative filtering and although there are services which offer recommendations based just on audio features, they often miss the human touch. Our project aim is to build a system that combines both social and waveform aspect of music and build a recommendation system that creates a playlist based on the likes and dislikes of the user. The core aspect of our system are the features of audio signals which we will further polish by using tags from last.fm dataset. For creating recommendations, we will use the work described here. [7]

The basic idea is to use simple techniques and constantly consider feedback to keep the process going in the right direction. User can keep liking or disliking songs even after the initial setup process. Key aspect is that the initial setup time should be as little as possible. An important thing to consider would be variance, since too little variance would cause boredom and too much variance might not be well suited to the taste of the listener.

2. METHOD

The basis of our approach would be waveform similarity beween audio songs. These songs would be assigned tags as per the spectral similarities and patterns which would be calculated as per the information extracted from the million song dataset. To overcome the mechanical barrier and to introduce a flavour of cultural and human touch we will also annotate these songs from the tags extracted from the last fm web api.

Once the songs are annotated, the role of the user and the recommender comes into play. The recommendation cycle follows the following work flow :-

1) The user is presented with various moods/activities(For eg:- Working-Out/Studying/Partying) from which he can choose the option most suitable for him. Alternatively if the user cannot find any option defining his current requirement, he would also be having an option to input a seed song by recording a song that suits his mood.

2) Once an option is selected/or seed is uploaded, our system will create a static playlist containing 5 songs based on the option choosen or based on the spectral similarities with the seed song and a candidate song closest to the seed song is played.

3) The next song played is the candidate song closest to any of the accepted songs. Using the minimum distance for recommendations from song sets that is proposed in [6].

4) For each candidate song, let da be the distance to the nearest accepted, and let ds be the distance to the nearest skipped. If da ¡ ds, then add the candidate to the set S. From S play the song with smallest da. If S is empty, then play the candidate song which has the best (i.e. the lowest) da/ds ratio. [7]

3. DATASETS AND TOOLS TO BE USED

We will use a subset of the million song dataset [2], and its associated apis to gather useful information about the songs for the purpose of algorithmic classification of the songs.

We will use the last.fm api [1] and its associated million song dataset [5] for the million songs dataset to gather social tag info.

Well use the spotify application [3] to play songs and the spotify api to create and modify playlists [4].

We will use Ionic framework to build the web based application and its mobile counterparts.

We will use Weka for machine learning aspect of the project and Apache Spark for data parallelism to achieve higher levels of efficiency in our algorithm.

4. PHASES

4.1 Phase 1 - Finalizing the approach, getting the dataset ready for use, Classification/tagging of songs.

The goal of this phase is to finalize the core algorithm to be used, validate that the tools decided are optimum for our use against the various other tools available, selecting, preparing and pruning the dataset to be used and getting it available on the server to be used for the project and classifying the dataset as per our need. Since we will be using a subset of the million song dataset, we might not get all the features we would ideally want to extract from a piece of music. We have to identify the constraints and define how to overcome them.

4.2 Phase 2 - Building the recommender, training and testing the classifiers

This is the core part of the project as we will be defining and preparing the models for our recommendation. we will be building the models on a crude level in this phase which would be optimized and polished in the later phases. The core approach remains audio-features based music similarity but we have to refine it further by including artist, genre and other tags. We will have a basic working model of the application by the end of this phase.

4.3 Phase 3 - Building the web application along with its mobile and wearable counterparts, integration of the interface and the recommender and deployment

Goal of this phase is to create a polished final product that works on both mobile and desktop platforms. Recommendation engine will run on a server and applications on all platforms will consume the web services deployed on the server to interact with the core recommender. Integration of the server part with the client part will also be done in this phase which will include deployment of the webservices on a suitable server and setting up the data parallelism to improve performance and turn around time. We will be using ionic framework to build the application and that way we can build applications for the web, iphone and android simultaneously.

4.4 Phase 4 - Testing, feedback and further improvements

In this phase we will test the application ourselves first and then put it to real world test by asking users to listen to recommendations generated by the system. Consider the reviews and if possible apply those changes to improve the application. We will work on various fine tunings in this phase in which might result in better and improved results in an efficient manner.

5. ROLES

Member	Role
Harpreet	1) Estimating Ground Truth (Social Tagging) 2) Building the core recommender 3) Product management
Ishmeet	1) Estimating Ground Truth (Feature Tagging) 2) Handling machine learning aspects 3) Enabling Data Parallelism
Harmeet	1) Handling Deployment on server 2) Developing mobile, wearable & web apps 3) Testing

6. TIMELINE

Phase	Deadline
Phase 1	March 8, 2016
Phase 2	March 25, 2016
Phase 3	April 4, 2016
Phase 4	April 10, 2016

7. REFERENCES

[1] Last.fm Web Services, available at. `http://www.last.fm/api`.

[2] Million Song Dataset, official website by Thierry Bertin-Mahieux, available at. `http://labrosa.ee.columbia.edu/millionsong/`.

[3] Spotify application, available at. `https://www.spotify.com`.

[4] Spotify Web API , available at. `https://developer.spotify.com/web-api/`.

[5] The Last.fm Dataset, available at. `http://labrosa.ee.columbia.edu/millionsong/lastfm`.

[6] Beth Logan. Music recommendation from song sets. In *In Proc ISMIR*, pages 425–428, 2004.

[7] Elias Pampalk Tim, Tim Pohle, and Gerhard Widmer. Dynamic playlist generation based on skipping behavior. In *Proc. of the 6th ISMIR Conference*, pages 634–637, 2005.

[8] Douglas Turnbull, Luke Barrington, and Gert Lanckriet. Five approaches to collecting tags for music. In *In Proceedings of the 9th International Conference of Music Information Retrieval*, pages 225–230, 2008.