Thoughts on music recommender systems for personalized radio stations

Andre Wiethoff
April 2015
streaming@exactaudiocopy.de

1 Introduction

As the amount of musical works grows steadily, it becomes very hard for a music lover to find (new) songs of which he would be fond of. Currently radio stations occupy a specific niche, they usually play songs that are liked by a specific group of listener (most of them serve various partitions of Pop content).

Online streaming services are booming, which allow users to create adaptive radio channels based on genre, band or song similarity, etc. which promise to generate a better experience than radio, as the song recommendations are tailored for the specific user. Most often that promise can't be kept, but some streaming services provide better recommendations than others. This analysis will try to find tools that could be helpful in improving the recommendation performance.

Please remark that this paper is not a research paper, as it does not live up to even very basic research standards. Citation would need to be much more precise and quite some sources are not listed at all. Further, claims are made, but no experiments have actually proved their correctness. It is just an analysis of potential problems in a kind of thought experiment and solutions that could be used on creating music recommender systems. This paper was written in only a few week's time, therefore the standard might not be as high as a research paper. Also the structure of the paper could have been further optimized as there are e.g. repetitions throughout the text, but the present form is due to the limited time spent on the creation of the paper.

Chapter 2 contains a very short overview over currently researched automatic music description algorithms. Most of the content in this chapter is work from various other papers available in the research community, but which are not completely cited. A small excerpt of the used research papers is listed in the addendum.

Then follows a chapter which puts the consumer into the center of the analysis and show the possible needs of such a user regarding personal radio stations.

Some possible implementation details, which are necessary to build a functional personal radio station, will be discussed in chapter 4.

Finally there will be some concluding comments and an addendum with some few external references.

2 Automatic music metadata retrieval

When trying to implement a recommendation engine, often the recommendations are based on algorithmic features of the music data itself. Other researchers use collaborate information or analysis of available (textual) metadata. The following sub-chapters will give a small overview of research fields which is not complete at all. Basically all presented information in this chapter is gathered from research papers, only some small parts are my own work.

2.1 Dynamics

Song matching could depend on the dynamic range of a musical piece. A romantic song followed by some death metal piece clearly doesn't fit the needs of a listener. Songs in a playlist should match the dynamic range of each other at least roughly.

Comparing two songs regarding their dynamic range could be a bit more difficult if one of the songs has applied a dynamic range compression, which could alter the dynamics dramatically. Therefore an applied compression should be detected (if possible) and be removed/compensated for comparison.

A possible algorithm for comparing dynamics will be discussed in the next chapter.

2.2 BPM detection

Likewise, songs in the same playlist should be of similar speed. In earlier times the Beats-Per-Minute (BPM) indicator was used to describe the tempo of a given song, but they realized that nearly no song has the same speed in all parts and some songs even do not have real beats (e.g. chill-out/ambient music). Even worse are classical pieces, where the speed changes very often radically.

Thus it would be possible to specify the parts of the given song which have the same speed and set a kind of BPM value for each part. On the other hand, a tempo indicator for the whole song would sometimes be more favorable (e.g. in order to calculate the tempo similarity of two songs).

Therefore I propose to use a vector of tempo information (each bin contains the share of beats of a given tempo). The first bins contain very slow beats, while the last bins receive information about very fast beats. Whether the beats (or for that matter: tone changes) occur at a quarter note, half note or eighth note doesn't matter, as only the beat intensity that occurs on such a beat will be added up in the bin. This should give a more intuitive approach to beats (and beat energy).

For that we need to set a valid range of BPM values, most probably values around 50-320 BPM will do. First of all it is important to normalize each song to a given loudness (e.g. using RMS). After that, we perform an autocorrelation on the derivative of the energy values of the song (which is 44.1 kHz downmixed to mono in the following examples), we only store correlation coefficients in the given range (50 BPM translates to $\frac{44100 \cdot 60}{50}$ samples and 320 BPM translates to $\frac{44100 \cdot 60}{320}$ samples). Therefore we only check the autocorrelation coefficients between 8268 and 52920 samples. In order to speed up processing we could use the STFT to calculate the autocorrelation efficiently and use a sound step size. The results are smoothed using an adaptive Gaussian blur. As we want to use equidistant BPM values, which differ greatly on both ends of the autocorrelation bins (e.g. between 50 and 60 BPM there are $\frac{44100 \cdot 60}{50} - \frac{44100 \cdot 60}{60} = 8820$ samples and between 300 BPM and 310 BPM there are $\frac{44100 \cdot 60}{300} - \frac{44100 \cdot 60}{310} = 285$ samples). Finally we sample e.g. 16 bins from the smoothed curve.

Similarly we are able to calculate a histogram of loudness. We determine the loudness of a small piece of sound (in dB) and storing it in a given bin (higher than the desired output resolution). Then the sample count of all bins needs to be normalized in order to reflect differences in song length. Afterwards a constant Gaussian blur is applied and sampled anew using 16 bins in this experiment.

Figure 1 shows the results of four different musical pieces.

"Nothing Else Matters" from Metallica is a metal ballad. Even though the main BPM is 142 (prior knowledge), the intensity (the result of the autocorrelation) is not very high, meaning that the beat is not very strong. It also has a smaller spike at around half the BPM, which tells that there are also beats generated by twice as long notes (and by a high autocorrelation in a distance of two quarter notes). The spikes around 300 BPM are most probably due to the vocals of the song. The loudness histogram tells that there are quite some low loudness parts (in particular the beginning of the song) and that the overall loudness is only medium.

In contrast the song "Welcome To Hell" from Sum 41 is a real fast (168 BPM) and aggressive song, the vocals are more or less only yelling. This is also deducible by the speed diagram; there is a real hard main beat at the given BPM and it even contains a part of very high BPM. The loudness histogram also reflects this, as it has the highest loudness of all songs.

Ambient music often is pretty soft without hard beats. We process the song "Summer Breeze" from Blank & Jones. I would guess it has a main beat at 74 BPM and also an auxiliary BPM at twice the speed. But the diagram shows that the beats at 148 BPM are more (or louder), therefore this song shows that a simple BPM detection algorithm could fail on detecting the correct BPM. The loudness histogram tells that the song has an average loudness, the loudness levels seem to flow within the song without too much up or down.

The last figures show a classical piece, the Allegro from the 5th Symphony from Beethoven. Exactly as expected, the BPMs are nearly a flat line. This is not due to the fact that there are no beats in the symphony, but that the speeds are varying so much that on average the piece contains the given beat half of the time and not for the rest. In fact, if applying the algorithm on an excerpt of one of the main themes of the Allegro, the BPM diagram shows much higher peaks. The loudness diagram is also as expected quite equally distributed over the different loudness levels.

In order to compare two diagrams you would e.g. use RMS of the differences of all levels to determine a kind of similarity in speed and dynamics.

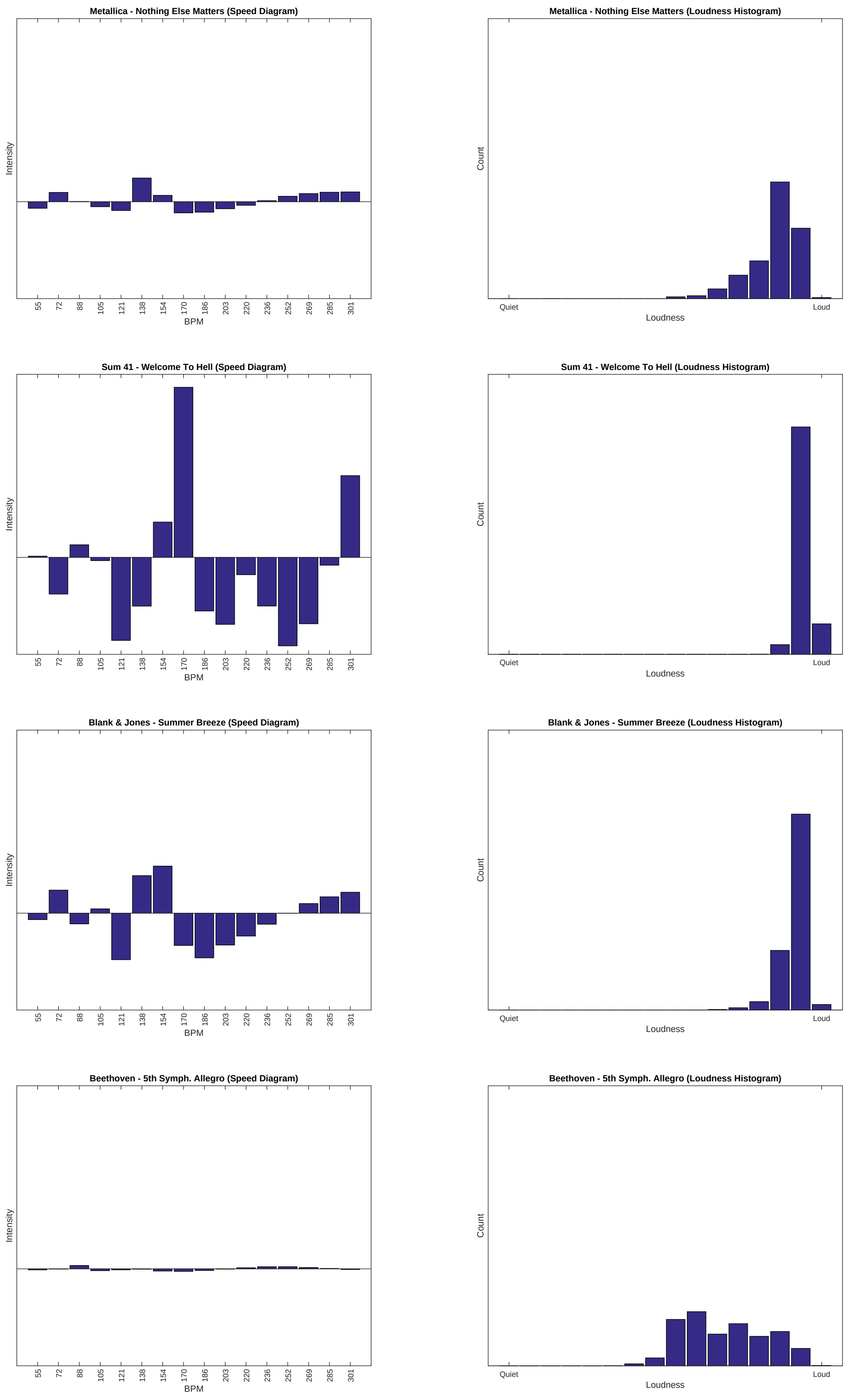

Figure 1: Speed and loudness diagrams of various musical pieces

Overall it might be a good idea to split the song into several (equally long) parts and determine the levels of each diagram for each part. Afterwards the average and standard deviation of each level might be a better parametrization.

2.3 Melody extraction

Extracting the melody of a song (or any other track) is quite difficult. Most music is a complex mixture of various sound sources, including harmonic instruments, percussion and vocals. There are algorithms that try to separate the mixture into single tracks (like Nonnegative Matrix Factorization and others). Anyway, none of them works perfectly (as even the ear is not always able to separate two sound sources perfectly). Therefore it is a hard task to identify the melody track and retrieve the duration and pitch of each note. If the melody could be extracted, it might be used for further analysis (e.g. to try to estimate the emotional perception of the song).

2.4 Chord extraction (major / minor scales)

A similar problem is to detect chords in a given song. Using the melody and the chords it is possible to determine the scale of the song. Using the scale information it would be possible to have a rough estimation of what kind of emotion is associated with it. Finding similar songs depending on their scales would be difficult, as the algorithms are far from perfect and the scale alone is only a weak indication of a songs emotional topic.

2.5 Instrument detection

An operative instrument detection would produce even more useful information. E.g. each genre often uses a specific set of instruments. Using the instrument and tempo/dynamics information we would be able to identify the songs genre with only little degree of uncertainty. Again, the extraction of instruments from a mix of instruments and vocals are not easy, as already described earlier.

Technically, the result of the instrument detection should be a value between 0 and 1, which describes how much and how intense the instrument is audible. A solo instrument should have a pretty high value (or marked special otherwise), while all unused instruments should be around 0.

Sometimes it will not be possible even to a human expert to differentiate several individual instruments from a mix. One possibility is to examine the whole file in order to find places where the detection for a given instrument is easier.

Another approach would be to check for instrument mixes instead of single instruments. The downside is of course that the number of different mixes increases very fast with each additional instrument.

2.6 Singer detection (male / female / duet / choir)

Also important is the detection of the vocals of the song. Of most importance is the gender of the artist(s) and whether it is a duet or even a choir (male/female/mixed). From the field of speech recognition there exist quite good algorithms to detect the gender of a speaker (and if there is no background music also from vocals).

Even the used language(s) can be detected to some extent, but for now it will not be possible to detect the lyrics automatically from a song (which isn't possible for some songs even for human listeners).

2.7 Genre detection

This is one of the main topics in automatic song metadata creation. Determination of the genre of a given song would help greatly to classify the song into one or more groups. Again, the result should be a value of 0 to 1, so that a song could be 80% Pop and also 50% Rock and 10% Metal.

For that, it is rather difficult to find a (large) set of songs for which the ground truth is known. This is what music experts must provide and with which a machine learning algorithm need to be fed. For such, nearly all outputs of other analysis tools (like instrument detection, speed, dynamics, etc.) are useful inputs.

Anyway, some genres are very hard to detect automatically, e.g. Children or Christmas. Modern Children songs often sound like Pop and also quite some Christmas songs are also of genre Rock or Pop. Even worse, there are songs which are not of Christmas genre, but perceived by the listeners in that way (e.g. "Last Christmas" from Wham!, as the story line takes place around Christmas time).

Experiments have shown that algorithms are better in telling whether the song is of a special subgenre instead of a main-genre. E.g. a machine learning algorithm has better results in predicting whether a song is Progressive Rock instead of telling whether it is Rock, as the differences between the various sub-genres are sometimes too large in order to find a good boundary for the main-genre. From the sub-genres it is then possible to assign a song to the main-genres.

2.8 Song lyrics analysis

The analysis of the lyrics is often helpful to determine the genre or the emotional effect. But there are three downsides with lyrics: they can't be generated from the audio data easily (at least for quite some time coming), some songs doesn't have any lyrics (e.g. most classical pieces, some soundtracks and ambient songs) and the lyrics might come in any language (even several intermixed). Therefore you would need to obtain lyrics from most of the songs in the library, but which aren't available to the streaming services. I would guess that at most 10% of songs have lyrics available.

But the lyrics could also be an indicator whether a song is a hit or not. For example the song "Demo (Letzter Tag)" from Grönemeyer is not very interesting when analyzing the musical structure and content (if you don't speak German, you should try to find out why the song is liked by many fans just by listening). But the lyrics of this song are poetry put into a song, making it difficult to analyze and to rate.

2.9 Similar bands analysis

In order to analyze how similar two bands are, you could use the audio data to determine various parameters and compare them. This is still very difficult, it is quite easier to use metadata which is provided by the music metadata services (like AMG) or mine this information from the Internet. E.g. Amazon has calculated all such information from user purchases. If you go to the Amazon web page and select an artist page (by clicking on the band name), Amazon proposes similar bands. For Metallica it e.g. proposes Iron Maiden, AC/DC, Slayer, Megadeth, Guns N' Roses, Anthrax, Judas Priest and Motörhead. For Queen it proposes Freddie Mercury, The Who, Genesis, Pink Floyd, Supertramp, Deep Purple, Montserrat Caballe and Thin Lizzy. The bad thing about such mined information is usually that the count of connected artists is usually limited; there are usually quite more than eight bands which are similar to a given band. Of course streaming media services could generate such information also on their own using information about what bands each user listen to.

2.10 Emotional metadata (arousal / valence)

One important aspect of a song is what emotional response it invokes in a given listener. The problem is of course that one song can produce different emotions for different persons, as each one always listens to the song in a given context. E.g. a song that you heard when first meeting your loved one will be perceived within that context. As it is impossible to automatically determine or even define the context of each and every user, it need to be enough to find some objective description of a song. It would be very helpful if we would know that a specific song is sad, happy or relaxing. Most research agrees to a 2D valence-arousal emotion space, which is often depicted as a circle. While the arousal of a song can be detected from the tempo, loudness level and timbre of a song quite well, the valence could be related to the used scale and harmony. But estimating good values for the valence of a song is still a difficult task.

Usually each quarter of the mentioned circle will receive three possible values (which should be only used as an internal description, as they often do not map very well to user made descriptions). With negative arousal and valence usually the values sad, bored and sleepy are associated, with positive arousal and negative valence the following terms are used: annoying, angry and nervous. For negative arousal and positive valence you would get relaxed, peaceful and calm. And for arousal and valence both positive, descriptions would be excited, happy and pleased.

Having such information would greatly help in providing good recommendations based on a given song, as all following songs could have a similar emotional response.

One additional thought, the arousal and valence parameters could of course change within a song, so a song which starts happy could end very sadly. It would be very difficult to map such a progress into metadata. Most often it will be helpful enough to know if the song has mostly a given emotional response.

2.11 Social metadata generation

Last.fm, one of the oldest player in the streaming business, is based on social metadata. The data (called tags, which are textual descriptions) have no special format. Any registered user is able to add a tag to a song, or increase the value of an existing tag. Additionally the user is able to like the song or to dislike the song. There are two disadvantages of this scheme, first of all the tags are not grouped or named regarding a specific scheme. Often two tags are telling the same information, e.g. some tags are just existing tags which are translated into a different language. Also there is no way to determine the group of the tag (whether tag is a genre tag, a band tag or something else). Further, the number and values of tags decreases quite fast as soon the song gets more unpopular, therefore the quality of the metadata is not the same for all songs. Usually there is no problem with users which introduce wrong or bad metadata into the system, as the value of their metadata would be pretty low.

Anyway, having metadata information specified by users is a great tool, as they can provide some higher-level semantic descriptions which cannot be generated by algorithms. For example whether a song is a ballad, in which language(s) the lyrics are or whether it is perceived as sad is often not provided by commercial metadata providers. Also the correction of genre information of a given song could help to own high quality metadata for the available songs.

2.12 Conclusion

All the described algorithms are actively researched, but none are already ready for production use. Also some other algorithms might be useful, like detection of Live recordings, etc., but aren't of great interest so far. Several years will pass until stable algorithms will be available for recommendation engines.

Therefore other information should be used in recommendation engines for now. But as soon as any of the research fields has results which produce information at least as good as human experts, these additional information should of course be integrated in the existing code base.

3 User centric recommendation approach

3.1 Musical knowledge of the user

The problem on interfacing with the users are the different levels of music understanding. While some people (e.g. musicians) can clearly tell what they like or dislike on a specific song (qualitative), the broad crowd will only be able to tell how much they like a song (quantitative). Therefore the general user interface should target to simpleness, but offering advanced controls to more advanced users. Therefore I will propose different user interface methods for creating playlists in the next section.

3.2 Ideal music retrieval system

So the basic question is what would be an ideal music retrieval/recommendation system as part of a streaming radio station?

As a minimum requirement all played songs in a playlist should be normalized to a given reference level, so that the user doesn't need to change volume after a new song starts to playback. If the user chooses to play a whole album, the normalization should be applied to the whole album instead of the individual song.

Additionally a kind of car playback mode should be implemented, which will be most helpful for classical pieces. Most radio stations will add some kind of compression to the broadcasted songs, as it will adjust parts with a too low loudness to a better audible level. Especially in noisy environments (car, train, etc.) this would help to enjoy the music without changing the volume constantly.

Of course it should mainly play songs that the user probably likes and the songs should roughly belong to the same genre. Also the speed and dynamics of the songs should not vary too much. All these points are also valid for over-the-air radio stations. In such commercial over-the-air radio stations, usually playlist managers will provide a harmonious mix of songs which are selected on the basis of the playlist managers' experience.

The ideal system should be as good as a commercial radio station, but could even be better as it will be able to learn the preferences of the listener. Therefore the first thing which could be improved using a streaming radio station is that every user can receive its own individual playlist, which is optimized regarding his preferences. By using machine learning and recommender systems it should be possible to propose only songs which the user likes with a high probability.

To help the user with the transition of a preferred over-the-air radio station to a streaming radio station, the user should be able to specify such a preferred radio station on the first registration of the service. Then ratings of a wide range of songs can be directly applied to the users' preference statistics. On the other hand a radio channel which offers the same kind of music as the appropriate over-the-air channel could be provided.

For that the service could get the song information from the over-the-air stations either by metadata ripping (most stations are sending the song title and artist with the actual music stream) or by using audio fingerprint algorithms. The service doesn't need to monitor the station 24/7, taking 10 seconds of audio every now and then will usually do in order to receive a good fingerprint result. Metadata are usually sent with the first package of a Shoutcast stream, so connecting from time to time will also deliver song information (in whatever format, as the texts transmitted on Shoutcast streams have no standard artist/title formatting). Using fingerprinting technology has an advantage over the mere collection of stream metadata, as it will identify a specific version of song more accurately, as it would be possible that there are several versions of a song which have the same title and artist (e.g. dance version vs. live version vs. radio edit version) and fingerprinting will determine the exact version of the given song.

Additionally, the user should be able to start a radio channel constructed from various information. One possible request would be to play songs within a given genre (main-genres and also sub-genres), in this case the service should propose a wide variety of songs within the genre. Using the user ratings, the service would adapt to the user's preferences fast. Another possibility would be to specify an artist or band; in that case songs from similar groups should be played. Anyway, the similarity (or neighborship) of artists should not be defined too tightly, as otherwise only songs from a handful of bands will be played. As a third possibility the user should be able to specify a song, on which basis he wants to listen to similar songs. This is a quite difficult undertaking, as the service doesn't know the reasons why the user proposed the given song. A good way to find out what the user expects from the service, tags of the songs could be displayed of which the user chooses one or several tags which are important for him regarding the given song. For this a perfect set of standardized tags for each song would be necessary. Some tags could be generated automatically using algorithms (see previous chapters), but some need human specification (from either users, music publishers or metadata providers, e.g. the genre Soundtrack could most often not be autodetected). As an example we have a look at the tags from last.fm for the song "Nothing Else Matters" from Metallica. It defines several genres (heavy metal, metal, rock, hard rock); further some additional descriptions (ballad, emotional, guitar, guitar solo, male vocalists, melancholic, sad, slow, 90s) which could be autogenerated in the future using algorithms and metadata from the songs. Anyway, generating such tags is a hard task and algorithms still fail on quite some songs, generating wrong tags. Further, some tags (e.g. Christmas, Children and Soundtrack) could not typically be detected by algorithms, as some background knowledge would be necessary. A fourth operation mode could be specifying a set of tags (e.g. Jazz, 90s, Slow) from a given set of tags. This is a more generalized version of the genre radio (as genre is usually also a tag of a song). This also requires a perfect standardized set of tags for all songs used.

At the time the user prefers streaming radio over the normal radio, any radio like extensions beside the music should be activatable by the user. The most important non-music extension would be news, configurable for when it should be send (e.g. each full half hour) and what to be send (world news, local news, weather, traffic, etc.). Also other informational shows like movie reviews (or, if preferred by the user, book reviews), etc. could be aired as an option. Therefore the user would be able to create a fully customized version of his radio, playing all the music he loves and only sending information he wants.

Theoretical extensions, which would be more science fiction than the other interface proposals, would be to generate playlists depending on the current time/date or location of the user. Also as the mood of the user changes from situation to situation, this could also be reflected in the playlist.

Additionally, an ideal service would also respond to relative change requests of the user (e.g. slower, more piano and sadder). Anyway, this is nothing which would be feasible within the next 5-10 years, as it not only require tags as flags, but tags as rational values (e.g. a speed value, a value of how much content of each possible instrument is in the song and an emotional value). Letting users vote about these values would be very difficult, as there is no practicable way to measure this objectively (it is hard to tell whether one song is really sadder than the other).

These proposals are just a wish list how an interface of an ideal recommender system would act like, but doesn't explain how. Chapter 4 will try to analyze how to find an ideal individual playlist with the techniques available today.

3.3 Acceptance of User Interface Design (GUI)

The interface for selecting a new radio station (playlist generation) should be as convenient as possible for the user, showing only options which are helpful for creating the list. Different modes, as described in the previous chapter, should be presented with as few clicks as possible. An advanced form could offer additional options like how distant the songs of the playlist may be regarding the specified song origin (depending on a given distance metric of two songs).

While playback, the buttons that will be most useful for the user are the "skip to the next song" and rating button(s). Of course the album cover, artist and song title should always be displayed.

Neither skipping of a song nor letting it play to the end should impose any rating information from the user about the song. Letting it play can't tell anything, because the user could have left the room and let the service play unattended. Skipping a song could have various reasons (e.g. user has heard the song somewhere else lately and doesn't want to hear it again right now, the user is currently in a different mood, etc.). Therefore it is not possible to reliably educe something from the skipping of a song.

One of the important questions is: should one or more rating buttons be presented? Usually at least one button is available, the "love it" button for marking songs that are preferred by the user. Most often also a "hate it" button is present to rate songs that the user strongly dislike. I would argue for a more differentiated system, which offers five states (best presented with thumb signs), thumb up, thumb medium up, thumb neutral, thumb medium down and thumb down. This needs to be a compromise between the user's ability to decide for a given rating (which will be more difficult if there are more possible ratings, e.g. an integer value between 0 and 100) and the strong wish that the user rates every song played by the service (which is only possible when the user will be able to specify his emotion about the song more detailed). As music is strongly subjective and emotional, no music analysis algorithms will ever be able to find a good set of songs for a specific user, as two users who love song A could have different emotions about song B (whereas an algorithm would always work deterministic). Only a dedicated recommendation system based on machine learning using rating data from a large number of users will have the desired effect. Therefore the machine learning system needs as many and as detailed data as possible for creating recommendations that a specific user will like. Offering five buttons seems to be too much to decide upon by the user at first glance, but the user need to think about it anyway if he chooses whether to use the "love it"/"hate it" button or not. Using five categories, any song of the highest two categories will be great to listen to for the given user, the third category should only be played for newly released songs (where no sufficient rating data is available), which perhaps need to be reheard several times (as a user will sometimes like a song better when he hear it more often). If a song gets rated twice, the second rating should always overwrite the first rating.

Quite some users own HIFI devices that connect to the streaming services. Even though some services offer family accounts, it is not easily possible to set the active user in the GUI via 1-2 clicks. Most often the current user needs to be logged out and the new one has to enter all his credentials anew. Therefore songs are probably often learned as preferred, which belongs to a different family member account.

4 Basics of a recommendation system

This chapter discusses some implementation details of a possible ideal recommendation scheme.

4.1 Volume normalization

In order to deliver all songs at a given volume level, it is at first necessary to determine the maximum loudness of each song in the system. Very simple techniques like peak detection will not show the desired results. It is better to use at least a RMS approach or even better the ReplayGain standard (see link in the reference section). Once the maximum loudness of each song is determined and stored in a database, it is possible to bring all played songs to a specified reference level (e.g. a song with -10 dB maximum loudness needs to add 4 dB for the whole song in order to end up at the given reference level of -6 dB).

The amplification is performed straight forward (a clipping prevention should be applied and afterwards some dither should be added). This can be performed in time domain, thus the speed should be sufficient even for a web player (unless the decoding occurs in hardware though, in that case the gain factor should be incorporated into the audio

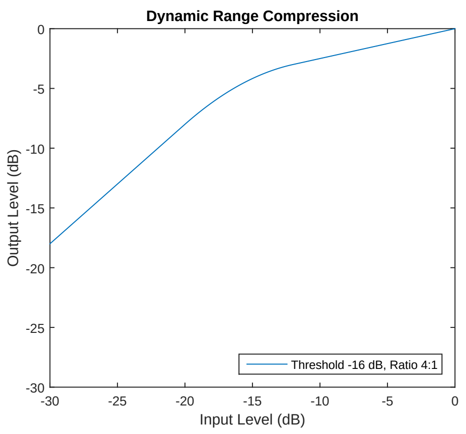

Figure 2: Dynamic Range Compression

file as metadata if the web player supports the extra metadata – or as an extra encoded audio file in all other cases).

4.2 Dynamic Range Compression

Most radio stations will apply Dynamic Range Compression (DRC) to the broadcasted material. Even though the dynamic range of a song will suffer, this has an advantage for the casual radio listener, which doesn't listen actively to the radio in front of an HIFI device or using headphones. The reduced dynamic range removes the necessity to turn the volume up and down constantly. E.g. a classical piece could have a very high dynamic range; parts with low loudness are intermixed with parts of high loudness. In a car, the listener needs to adjust the volume, otherwise the loud parts would be too loud or the quiet parts would be too quiet (or even inaudible). Playback in the car would be one extreme, but even working while listening to the radio would be difficult, as the ambient noise will prevent the reception of the quiet parts. Therefore it might be a good idea to offer the possibility to have the dynamic range compressed (in one or two steps), or to playback the audio as is (as preferred by the listener and/or used device). As this processing would take quite some processing power, a good solution would be to store each song in two or three versions, one original and one or two with reduced dynamics (low compression and high compression). In figure 2 an example of a compression curve is shown, even though the best threshold position, ratio and gain for a specific use needs to be determined. On playback the selected version of the song needs to be streamed. As many newly released songs already have applied a high compression, an auto-detection of the original dynamic range of a song could be helpful in order to create an adaptive scheme for compression.

4.3 Genres

Automatic genre detection is still a quite difficult field of Digital Signal Processing. Even though some results have been obtained, it is still not robust enough to use unattended in a production environment. Even more, some genre information can only obtained by additional metadata from the music company or from other human interaction (e.g. Christmas, Children and Soundtrack). Therefore it will not be possible to get around retrieving the information from commercial music metadata databases, which specify one (or sometimes several) tags for the genre of the song. Quite often the given genre of a song is just the most used genre of the artist or the genre of the album, so the genre information could be wrong for individual songs. Furthermore, these tags are flags, either the song is of that genre or not. Algorithmic determined genres could provide a range e.g. between 0 and 1, where 0 denotes that the song is not of the given genre at all and 1 that the song is of the genre. That way, a song could be of several genres, each of them more or less. E.g. a song can be 80% Jazz, 30% Pop and 10% Rock (the values does not need to add up to 100%, for that the values need to be normalized first).

Genres are a quite important part of the playlist generation, as users usually don't want for stray too far away from the originally specified genre, someone who specified pop doesn't necessarily want to have classical music in his playlist.

4.4 Band similarity

Most recommender systems offer radio stations or playlists which originate from an artist or band. Most often songs of that band and related bands are played. Related bands are bands which perform similar music. To determine related bands, it would be quite difficult to identify them by comparing the similarity in their music (as it is rather difficult to define a similarity measurement of songs). Therefore most companies will use pre-calculated sets of artists (e.g. defined in the metadata of AMG or on the webpages of Amazon), which are either based on collaborative filtering (if many users like two bands, they will most probably be related) or manually specified by some experts.

Sometimes such information is obtained by data mining on the internet to see how often two bands are specified in a context (e.g. Amazon). The band similarity can also be directly determined by performing an analysis on the rating information of the streaming service if a good recommendation engine (machine learning) is implemented. But in this case, the similarity can be determined based on a specific song of the band instead of the artist or band.

4.5 Repetition rate of songs

Of course songs should not be re-played within quite a while. The repetition rate should orientate at the number of songs played between the last playback time and now. Also the rating of the given song should be intermixed, so that highly rated songs are played a bit more often than mediocre rated songs (and poorly rated songs should not be played at all).

So we can use e.g. a linear or Gaussian transition of probabilities from 0 to 1 for a given number of songs that played between two occurrences of the some song. From a given number of songs in-between (probably around 10000), the probability will stay at 1.

4.6 Tags

As we have seen in chapter 2, it is difficult to provide tags for the songs. There are more important tags than others, like genre (multiple genres if necessary). For now it is not feasible to generate rational values using algorithms, these will still take several years of research, even if there are great advances in the last years. Some few values can already be determined right now, like song speed and dynamics (see last chapter). So even if it would be nice to have a full set of tags, this can only be performed by using a flag for each tag and let (expert) humans decide the setting for each tag. If enough money could be provided, this information could be created manually for the important 10% of all songs (these that actually will be played in every genre, etc.) by using services like Mechanical Turk from Amazon. This would result in the advantage of having a standardized set of tags. Last.fm offers user generated tags, but quite often the tags are doubled by different spellings or translations. Therefore it is difficult to make use of such tags, even more as the data is only a pile of tags and thus the category of the tag is unknown (genre, decade, emotional state, etc.).

Finally, some other types tags can be automatically generated by the metadata of the song, if available (e.g. the release date forms the 80s, 90s, 00s, 10s tags, etc.).

Anyway, having a set of (standardized) tags would be very helpful to communicate with the user about his preferences. E.g. on starting a radio channel basing on a specific song, the user could select the tags associated with the song that resemble the (subjective) important features of the specified song and thus improve the recommendation result.

4.7 User recommendations

As in every field of art it is nearly impossible to let a computer predict how a user responds to a specific piece of art, as art is highly subjective. No analysis algorithms in the next 20 years will be able to exactly predict whether a user likes a song or not. As described in chapter 3 we therefore need a scheme in which users are able to specify ratings for songs, the more (and more detailed) ratings the better the result that a recommendation engine can deliver.

If we have rating information for a song (either directly via input from the user or via the result of a recommendation algorithm), the song should be played next with a probability depending on its rating. If e.g. the rating information is stored as values between -2 (hate it) and 2 (love it), the play probability would be $\frac{RI+alpha}{2+alpha}$ if the value is larger than zero and zero otherwise (with alpha of << 1). At the beginning all songs will have a rating of 0, thus every song will have the same probability of alpha to be chosen next.

4.8 Selection of songs

The most important step for choosing songs of a playlist is to filter out songs that the user most probably doesn't want to hear. This is exactly what radio stations do, they play hits all the time to keep the listener in the channel as long as possible in order to have the possibility to also send them advertisements. Therefore, the user expects from a radio (over-the-air as well as streamed) that every song is a potential hit and has the chance to be liked by him.

Most streaming services have statistics on how often a song is played (and eventually liked). If this list is sorted by their playback count (this should be applied to a genre/tag filtered song list - on band similarity filtering, each band should be processed

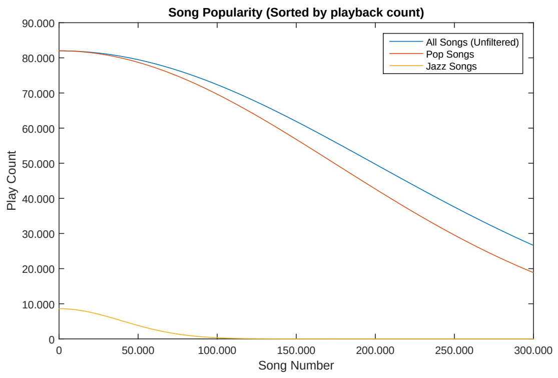

Figure 3: Sorted playback counts of songs

separately), usually a half of a bell shaped curve will show up, which looks like a Gaussian distribution.

Please see figure 3 for an illustration of such curves (artificially created without any actual data) and figure 4 for an example for the band Supertramp (retrieved from the Simfy application - it wasn't possible to retrieve a list for a specific genre or for similar bands). If we assume the average at song position 0, we are able to calculate the standard deviation. The "68-95-99.7" rule tells us that 68% of all playbacks are within one standard deviation and 95% of all playbacks are within two standard deviations. Therefore we want to find a point (a song number in the sorted list), up to which we will play songs. All other songs, which belong to the so called Long Tail, have in sum around the same number of playback counts, but are only heard by very few people. The amount of songs which should be taken into account could be a parameter of the user, perhaps he wants to explore more unknown music (and thus not listening to the top hits at all). Anyway, from the 20-30 million songs that are often available for streaming, probably far less than 1 million songs should actually be used as a basis for playlist generation.

In the Supertramp example, we roughly calculate a value of 9,7 as standard deviation (without having information about additional songs behind position 30). This means that the first 10 songs are within one standard deviation (and this are roughly 68,3 % of all played songs). The first 20 songs are played roughly at 95,4 % of the time.

As another sight on the problem, we can just use the (normalized) number of playbacks for a given song as probability on whether the given song should appear in the playlist.

The song play count should only be increased if at least 80% of the song are played and the user rates

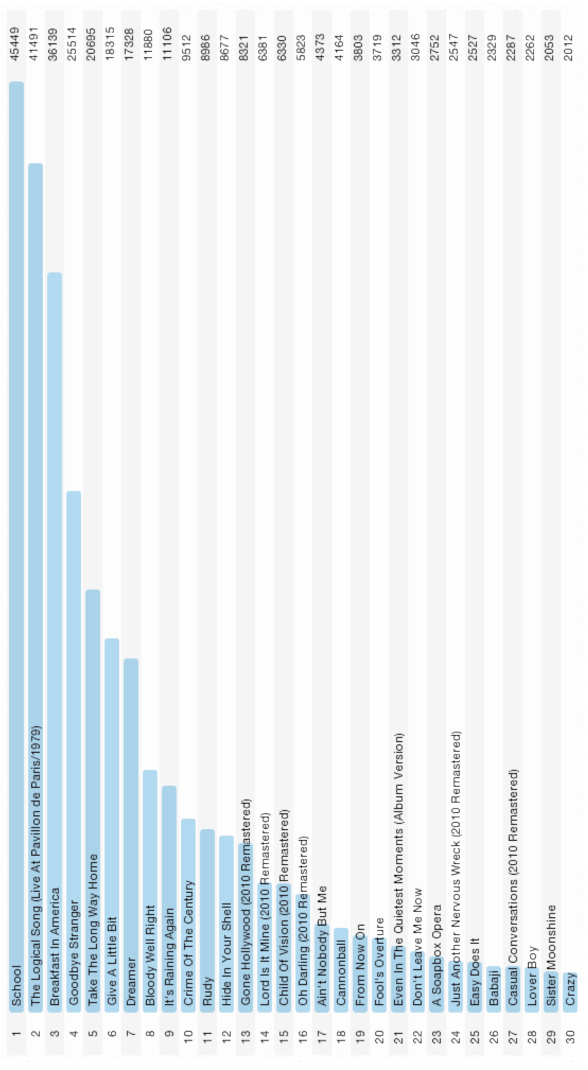

Figure 4: Sorted playback count for Supertramp

the song at least with 0 (if using a rating scheme from -2 to 2). It should not change if it just played through, but it should be increased if the song is explicitly searched for or specified by different means (e.g. by using a personal non-public playlist).

Finally, it might be a good idea to capture the playback count over a specific interval instead of starting from the beginning of the service. A hit from today could be out of fashion within some months. Therefore checking only the last few months of logs would prevent playing songs that no one will hear anymore.

4.9 Randomization

If the recommendation matrix hasn't changed (which will be updated probably every night, if not a kind of online update mechanism is used), the results would be always the same if not using any kind of randomization. The randomization is part of the probabilities given for the repetition rate of songs and selection of songs.

Therefore a random variable will model the different outcome for each new playlist generation of the same category (genre, etc.).

Basically introducting a random number e.g. in the interval $[0.8; 1]$ could produce the necessary randomization.

4.10 Paid song promotion

Finally it would be possible to promote songs of music publishers to users which will likely also love the promoted song. Then the music publisher will only be invoiced for fully played songs (>80 %) which are either not rated at all or not rated worse than average (comparable to CTR from normal web publishing advertisements). Of course impression based invoicing (CPI) would be possible, too.

The implementation would be based on an artificially created user which likes a number of songs which are similar to the promoted song. Then the promoted song will receive a rating which has a rating which is higher than normally possible to accommodate the problem that a single user will influence the recommendation only very little. The exact value depends on the size of the matrix and thus isn't constant.

4.11 Conclusion

With the described minor modifications most streaming services could be improved to produce better recommendations for their users. In the future more exciting possibilities will arise to improve the recommendation results even further. Until then, only basic algorithms as those proposed are available. Anyway, even these will improve the majority of streaming services so that they are able to become true competitors to over-the-air radio stations.

References

[1] Barbedo, J.G.A. and Lopes, A. 2007. Automatic Genre Classification ofMusical Signals. EURASIP Journal on Advances in Signal Processing

[2] Knees, P. and Schedl, M. 2013. A survey on music similarity and recommendation from music context data. ACM Trans. Multimedia Comput. Commun. Appl. 10, 1, Article 2 (December 2013), 21 pages.

[3] Schedl, M. 2013. Ameliorating Music Recommendation. MoMM2013, 2-4 December, 2013, Vienna, Austria

[4] Schedl, M., Pohle, T., Knees, P. and Widmer, G. 2011. Exploring the music similarity space on the Web. ACM Trans. Inf. Syst. 29, 3, Article 14 (July 2011), 24 pages.

[5] Yang, Y.-H. and Chen, H. H. 2012. Machine recognition of music emotion: A review. ACM Trans. Intell. Syst. Technol. 3, 3, Article 40 (May 2012), 30 pages.

[6] Ng, A. Machine Learning, Stanford University. https://www.coursera.org/learn/machine-learning

[7] Robinson, D. ReplayGain, Wikipedia. http://en.wikipedia.org/wiki/ReplayGain

[8] Mechanical Turk. A marketplace for work, Amazon. https://www.mturk.com/