uc3m | Universidad Carlos III de Madrid

University Degree in Audiovisual Systems

2017-2018

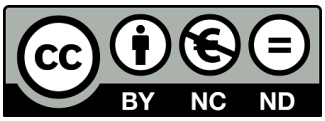
Bachelor Thesis

"Music Recommender Systems. Proof of Concept"

Ruth Borque Gallego

Advisor: Carmen Peláez Moreno

Madrid, October 2018



Abstract

Data overload is a well-known problem due to the availability of big on-line distributed databases. While providing a wealth of information the difficulties to find the sought data and the necessary time spent in the search call for technological solutions. Classical search engines alleviate this problem and at the same time have transformed the way people access to the information they are interested in. On the other hand, Internet also has changed the music consuming habits around the world. It is possible to find almost every recorded song or music piece. Over the last years music streaming platforms like Spotify, Apple Music or Amazon Music have contributed to a substantial change of users' listening habits and the way music is commercialized and distributed. On-demand music platforms offer their users a huge catalogue so they can do a quick search and listen what they want or build up their personal library. In this context Music Recommender Systems may help users to discover music that match their tastes. Therefore music recommender systems are a powerful tool to make the most of an immense catalogue, impossible to be fully known by a human.

This project aims at testing different music recommendation approaches applied to the particular case of users playlists. Several recommender alternatives were designed and evaluated: collaborative filtering systems, content-based systems and hybrid recommender systems that combine both techniques.

Two systems are proposed. One system is content-based and uses correlation between tracks characterized by high-level descriptors and the other is an hybrid recommender that first apply a collaborative method to filter the database and then computes the final recommendation using Gaussian Mixture Models. Recommendations were evaluated using objective metrics and human evaluations, obtaining positive results.
Key words: Recommender System; Music Recommendation; Music Information Retrieval; Collaborative Filtering; Gaussian Mixture Models.

Acknowledgments

I would like to thank all the people that helped me on my way. First of all thanks to Carmen, who helped me and motivated me during all the process. Thanks to my family for supporting me and giving me all their love. Thanks to Fran for being my inspiration. I am the luckiest person in the world for sharing my time with you. Thanks to my friend Clara for helping me whenever she can and spreading joy wherever she goes. And finally thanks to all the people that completed the survey.

Contents

List of Figures

List of Tables

1. Introduction

This bachelor thesis aims at designing, developing and testing a music recommender system. In the course of the development process different alternatives were analysed and evaluated. Although the initial intention was to obtain a unique system, experimental results led that finally two different systems were proposed.

The first system is correlation-based, proposed for applications with enough computing power to obtain recommendations as fast as needed or for situations were time is not an important limitation. The second one is an hybrid system, composed by a collaborative filtering method and a content-based system that models the playlist with a Gaussian Mixture Model.

The document is structured in the following way:

- First in Chapter 1 an introduction about the context of music recommendation is explained, including the regulatory framework and socio-economic environment. In that chapter the economic information of this particular final thesis are also detailed.
- Chapter 2 and Chapter 3 contain the state of the art in recommender systems in general, music representation and music recommenders in particular .
- In Chapter 4 and Chapter 5 the proposed recommender systems are presented, as well as the experimental process that justifies the decisions made during the development of the systems.
- Final conclusions are stated in Chapter 5 as well as the unexplored alternatives that are proposed as possible future work.

1.1. Motivation

The main objective of a music recommender system is to ease users finding new music that match their personal preferences at each moment. Playlists are lists of music pieces and songs that share some characteristics. As tracks in a playlist are intended to be listened sequentially we can assume that they will probably be listened in the same context. This simplifies a big problem of music recommenders that need to use user interactions or other sources of information if they want to provide recommendations adapted to the preferences of the user at each moment. One person could like different music styles but will not enjoy all of them in the same situation.

Also music streaming platforms offer considerably big music databases to the user that can be listened instantly. This increases the need of the users to organize their personal collections according to some criteria, which can be used to make quality recommendations that the user enjoy and find useful to discover new music. The size of the catalogue of these services may overwhelm users. Recommender systems can help find music they like without doing an extensive searching.

Music recommendation has become a hot topic for both academia and industry. RecSys Challenge 2018, which is the annual challenge that takes place in the context of the ACM Conference Series on Recommender Systems, focused on music recommendation specifically on automatic playlist continuation. As part of the challenge Spotify released a large and public dataset of playlists created by real users, which is very useful for research on this topic because it can be used to improve recommendations for the new listening habits of users.

Music recommendation has evolved in recent years due to that increasing interest in the topic. Music recommenders nowadays are able to provide successful recommendations very often but there is still room for improvement. In this thesis different recommender systems approaches are applied to a database of playlists to explore the possibilities of those techniques in that specific scenario.

1.2. Socio-economic Environment

1.2.1. Socio-economic impact

There are mainly two agents affected by music recommender systems: the music industry and music aficionados. Music recommender systems are typically useful in music streaming services. With the raise of music streaming platforms such as Spotify, Apple Music, Pandora or Deezer, music aficionados can explore and listen music databases that contain most of the pieces and songs that have been recorded in the last century. Some of the databases are freely accessible for users of the music service and others require to pay a certain amount of money for accessing. Money is no longer an impediment for listening the music we want at each moment. Also the effort needed to listen the preferred music is a minor issue, since most of the times it is only necessary to do an easy search of the song we decide that takes a few seconds. But such a cultural enrichment can be missed due to the choice overload. In that context music recommenders have the ability to help users by filtering the database according to their music tastes. Playlists and recommendations together can also be useful when a user is occasionally interested on a music style that does not correspond to the personal preferences so the user do not want to be recommended that type of music in the future.

As the number of users of those streaming platforms increase, artists have the opportunity to be listened by people around the world. Their music can be recommended to users that will probably like it. For not so popular artists it means their music will be listened by users that otherwise would never have known them. Small groups or musicians can reach a level of popularity that would have required more expensive promotional campaigns or marketing strategies. However depending on the recommender system used by the platform, songs from not well-known artists can be in a disadvantage position compared to popular songs in terms of possibilities of being recommended. Music publishers, distributors and companies related to music marketing could be also interested in recommendations because it can contribute to increase the popularity of music without investing as much money as in a promotional campaign. Music will be directly recommended

to the group of user that are more likely to be interested, so it can serve as an automatic market segmentation.

1.2.2. Budget

The development process of the project generated several costs:

- An on-line course about recommender systems in coursera was taken.
- There were weekly meetings with the advisor during 9 months that required a 35 Km car trip to the Universidad Carlos III School of Engineering.
- A computer was used for the whole process, which consumed much electricity. It was used for research on the topics of the project, for programming all experiments presented in Chapter 5,other tests and writing the memory document. Experiments required a lot of time (some of them even several days) so approximately the computer was used 1180 hours.
- Stationery for taking notes and printing relevant papers discussed during the weekly meetings.

  Considering the circumstances stated before, an approximation of the total budget is 319 €.

Concept	Price (€)
On-line course	40
Travel costs	238
Electricity costs	41
Total	**319**

Table 1.1. Approximate total budget of the bachelor thesis.

1.3. Regulatory Framework

Music recommenders are affected by some legal concerns, related to music royalties, users privacy and patents.

If the analysis of the audio signal is used for computing recommendations music royalties are paid first to get copyrighted music. Although there is royalty-free music, normally music intellectual property is owned by a person or a group people. Usually Intellectual Property and copyright royalties are managed by specialised organizations. A licence is needed in order to provide users the music audio. In Spain the most important institution is SGAE[1] and they specify several fees depending on the service provided: if downloads are available to the user or if the system is integrated in a music streaming service. In the first case there are different fees depending on the number of downloads. In the second case monthly fees depend on several characteristics of the platform, that can be summarized as follows:

- Use: commercial or non-commercial.
- Number of monthly service visitors.
- Number of users that can play music on-line and the number of users that can play music off-line.

Another issue that concern recommender systems is the data protection regulation. The General Data Protection Regulation[2] (GDPR) came into force in the European Union in 2018 and it regulates the processing of personal data. If the recommender system need to process private data of users, affirmative consent of the user is required. The user also has the right to erasure. On the other hand in the EU there is also free flow of non-personal data, so databases that do not contain personal information of users can be freely distributed.

[1] http://www.sgae.es

[2] https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=OJ:L:2016:119:TOC

Recommender systems, and in particular of music recommender systems are patentable. According to the European Patent Convention if the system is susceptible of industrial application and is innovative it can be patented. There is a wide range of music recommender patents, for example a biometric-based music recommendation or music recommendation using emotional modelling.

2. Recommender Systems Overview

The large volume of available data on the internet is overwhelming and it grows everyday. Some filtering is needed for the relevant information to be useful. The main objective of recommender systems is to filter a big amount of data and provide a recommendation that might be of some interest to the user.

Recommendations can be personalized if any information of the user is used in the filtering process, or it can be non-personalized if no individual data of the user is taken into account. Deciding whether to use one or the other depends on the situation. For example in the case of a new user with no associated information, it is only possible to make non-personalize recommendations. On the other hand with active users it is possible to construct a profile that models their personal preferences, so in that case a personalized recommendation could be made.

To make a good recommendation at a certain moment to a specific user is very difficult because apart from the music itself there are other factors that may influence preferences. Some of them are quite static, like cultural or socio-economic factors, for example age, country or gender of the user. But other things that affect what people want to listen are more varying,like mood, weather, context or the activity the user is doing. Music is listened in many occasions with different purposes: motivating on sports, mood regulation, as background music on a friends meeting, etc.

Music recommendation has some particularities that make it different from similar systems, like book, film or products recommendation [4] . One important singularity is the size of the database. The size of music streaming platforms' catalogues are in the range of tens of millions, while catalogues of films and series are typically up to tens of thousands. That makes scalability an important issue. Another particularity is that usually songs have a duration of around 3 minutes and are listened sequentially more likely than one at a time. In contrast films, series and specially books take more time to the user to skip to another one. Music

recommendation is also more dependent to the context compared to the other mentioned systems, a user may prefer completely different types of music depending on the situation and might not want to listen that song in other contexts.

In Figure 2.1 a general classification of recommender systems is illustrated. The main three categories are:

- *Collaborative-Filtering Systems*. They focus on the interactions of users with the items (like ratings or personal lists).
- *Content-Based Systems*. They use characteristics of the items and their similarity to build preference models of users. The main goal of this type of recommender systems is to find the common characteristics of items liked by a user (items that received a good rating) and then recommend new items with that characteristics to the user.
- *Hybrid systems.* They combine methods from the previous systems to benefit from their advantages.

2.1. Collaborative Filtering

The basis of this type of recommender systems is that users with similar tastes would like similar items. Their aim is to predict how much a user will like an item and according to that select some items to recommend. In a typical scenario of a CF system the main elements are a set of items and a list of users. Traditionally the collaborative filtering problem was mainly focused on *rating prediction*. In that case preferences can be expressed explicitly by consciously voting items, but sometimes implicit votes are inferred from their behaviour. For inferring that implicit votes, actions of the users are interpreted in terms of what they show about the user's preferences. If there is a misunderstanding of the meaning of the behaviour of the user votes may not correspond to the actual preferences of the users and it will make the recommendation worse.

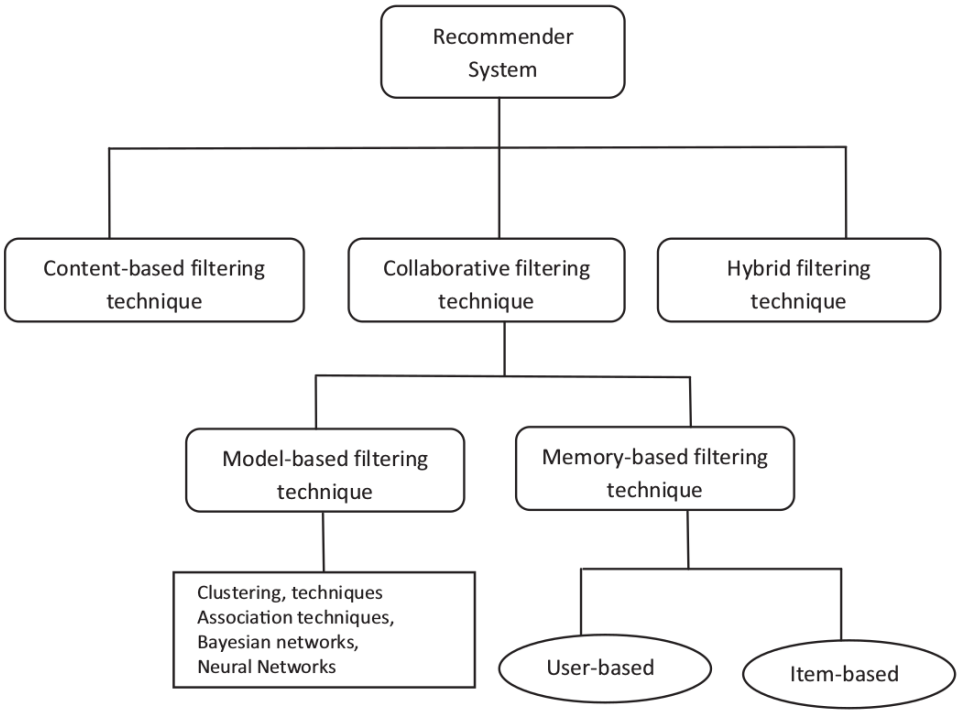


Fig. 2.1. General taxonomy of recommender systems. Source: [1]

Reacently collaborative filtering with *binary*, *positive-only data* is acquiring importance. Systems with binary, positive-only data aslo work with explicit and implicit feedback. In this case an example of explicit feedback could be 'likes' on a social network and implicit feedback could be the videos watched by the user. Rating prediction has become less important due to several reasons [5]:

- In systems that work with users ratings, collecting data requires that users make the effort of rating items.
- Correlation between users behaviour and their rating is not as high as expected because users tend to give higher ratings to items they think they should consume (e.g. users might rate higher a classic book of world literature than a superheroes comic even when they prefer reading the second one).

- In many cases only the higher predicted ratings are used because they would match users preferences better and at the end low ratings are irrelevant. But in rating prediction high and low ratings are computed with the same accuracy.

Users are characterized by a vector of votes, but depending on the system this vector can be a binary vector (that expresses if the user is interested in an item or not) or a non-binary vector filled with votes in other numerical scale. Typically there are missing votes in users' vectors. One important issue in some CF systems is how to interpret that missing information. For example: in a system in which votes of the users are inferred from the times they click on items, a user might not click on an item because he/she is not interested in that or because it is unknown to him/her.

Typically datasets in collaborative filtering are very sparse, which makes the problem more challenging. CF algorithms can be categorized into two subgroups [6] : memory-based methods and model-based methods.

2.1.1. Memory-based algorithms

The database is used to make recommendations based on similarities between in-memory users or items. The algorithm predicts the rating of a target user from the votes of other users. There are two different approaches: if similarity is computed by rows of the rating matrix (by users) it is a user-user memory-based system, if similarity is computed by columns (by items) it is an item-item memory-based system. In Figure 2.2

There are three important aspects that should be taken into account when designing a collaborative-filtering system: normalization of ratings, similarity computation, weighting the individual contributions and finally the selection of neighbours.

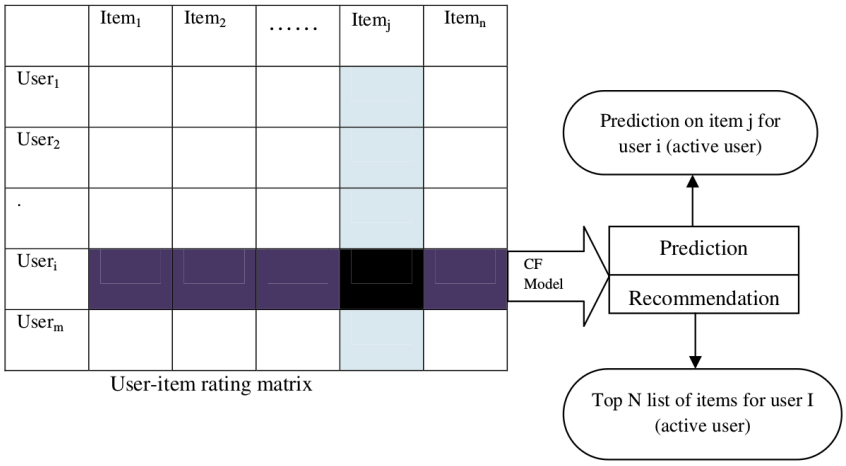


Fig. 2.2. Diagram of a user-based collaborative filtering system recommendation process. Source: [1]

Rating Normalization.

Different users may have different criteria when evaluating an item even when they have similar tastes. For example if we consider two users with similar preferences, one user could be inclined to assign higher ratings than the other. So there are two problems to solve:

- *Shifted average ratings*. As in the mentioned example, users may have different criteria when evaluating an item.
- *Different rating scales*. Some users tend to assign more extreme scores than others.

To account those factors a widely used option is to normalize for mean and standard deviation to obtain their z-score. So instead of using directly users ratings, similarities are computed from their normalized version. The normalization of users ratings r_i for the standards deviation and means to obtain the z-score is given by:

$$\mathbf{z_i} = \frac{\mathbf{r}_i - \bar{\mathbf{r}}_i}{var(\mathbf{r}_i)} \tag{2.1}$$

Let I_j be the set of items that a user j had rated, so the mean rating of that user is:

$$\overline{\mathbf{r}}_j = \frac{1}{|I_j|} \sum_{i \in I_j} r_{ji} \tag{2.2}$$

The predicted rating of an item i from a set of n items taking N users into account for the calculation can be expressed as a weighted sum as follows:

$$r_{ti} = \overline{\mathbf{r}}_t + \sum_j (r_{ji} - \overline{\mathbf{r}}_i)\, \omega_{jt} \tag{2.3}$$

Where r_{ti} is the prediction of the rating of the target user t over item i, $\overline{\mathbf{r}}_t$ and $\overline{\mathbf{r}}_j$ are the average ratings of the target user and a user i respectively, r_{ji} is a vector with the ratings of item i of the other users and ω_{jt} is the weights that represent similarity.

Similarity Metrics and Weighting.

There are several metrics that can be used to calculate the similarity between vectors, like *Pearson Correlation Coefficient*, *Cosine Distance* and *Adjusted Cosine Distance* [7].

- *Pearson Correlation Coefficient (PCC)*. It was first proposed in [8] to compute similarities between users in the context of the GroupLens project, that was very important for the CF research. The correlation coefficient goes between +1 and -1. The closer to +1 the correlation is the more similar their preferences are. A coefficient close to 0 means there is a little correlation between the users and a negative correlation can be interpreted as they have opposite tastes but it doesn't give information on how different they are, according to some experimental studies. The Pearson correlation between the target user t and a user i is defined as:

$$\omega_{t,i} = \frac{Cov_{t,i}}{\sigma_t\, \sigma_i} = \frac{\sum_j (r_{tj} - \mu_t)\, (r_{ij} - \mu_i)}{\sqrt{\sum_j (r_{tj} - \mu_t)^2 \sum_j (r_{ij} - \mu_i)^2}} \tag{2.4}$$

The item-based version of the Pearson correlation coefficient of the queried item k and another item j is:

$$\omega_{k,j} = \frac{Cov_{k,j}}{\sigma_k \, \sigma_j} = \frac{\sum_{u \in U_{kj}} (r_{uk} - \mu_k) \, (r_{uk} - \mu_j)}{\sqrt{\sum_{u \in U_{kj}} (r_{uk} - \mu_k)^2 \sum_{u \in U_{kj}} (r_{uj} - \mu_j)^2}} \tag{2.5}$$

Where U_{kj} is the set ot users that rated both items, μ_k and μ_j are the mean rating assigned to items k and j, r_{uk} and r_{uj} are the ratings that user assigned to the items.

- *Cosine Distance (CD)*. It measures the cosine of the angle formed by the rating vectors. The cosine distance between the target user t and a user i would be:

$$\omega_{t,i} = \frac{r_t r_i}{||r_t|| \cdot ||r_i||} = \frac{\sum_j r_{tj} \, r_{ij}}{\sqrt{\sum_j r_{tj}^2} \sqrt{\sum_j r_{ij}^2}} \tag{2.6}$$

  where denotes the dot-product of the vectors.

- *Adjusted Cosine Distance (ACD)*. In item-based systems the CD has a drawback because of the different evaluation criteria of the users so ACD subtracts the mean rating assigned by the user, μ_u. The final coefficient is defined as:

$$\omega_{k,j} = \frac{\sum_{u \in U_{kj}} (r_{uk} - \mu_k) \, (r_{uk} - \mu_j)}{\sqrt{\sum_{u \in U_{kj}} (r_{uk} - \mu_u)^2 \sum_{u \in U_{kj}} (r_{uj} - \mu_u)^2}} \tag{2.7}$$

Neighbours Selection.

The number of neighbours and the criteria used for their selection affect directly the system's performance. In big recommender systems the number of users can be too large to store in memory all users similarities. So in some cases it would be convenient to pre-filter the user database to reduce the number of candidates that will be taken into account in the recommendation. The three main methods that can be used separately or combined are [9]:

- *Top-N filtering*. By this filtering method only the N nearest-neighbours and their similarity weights. Parameter N must be chosen to be large enough to not lose too much variety but small enough to have an efficient system.
- *Threshold filtering*. All neighbours that have a similarity measure over or equal a certain threshold *th* are kept. This method is more flexible than the previous one but the chosen value must suit the systems requirements.
- *Negative filtering*. Negative correlations indicate that users belong to different 'groups' in terms of preferences but they don't give useful information for the recommendation. According to that, users with negative correlation could be discarded without worsen the system's performance.

2.1.2. Model-based algorithms

The database is processed to learn a model of preferences of a user based on previous ratings or purchases, which then predicts ratings of unrated items. The model can be built from various machine-learning or data mining techniques, some of the most used ones the following [1] :

Clustering.

Given un-labeled data, clustering techniques partition data into a set of meaningful sub-clusters. When the clusters are computed, recommendations for a user can be made by averaging ratings of users that belong to the same cluster and taking the items that have higher mean ratings. Many clustering algorithms try to minimize a function that measures the quality of the clustering, so it is an optimization problem. The optimization problem is computationally difficult so many algorithms use heuristics (e.g. k-means algorithm ending in local minima).

One of the most common clustering methods is *K-means* [10] . It is an iterative

process to minimize:

$$E = \sum_{i}^{K} \sum_{x_n \in S_i} d(x_n, \lambda_i) \tag{2.8}$$

where K is the number of clusters, S_i is the ith cluster of items which centroid is λ_i and $d(x_n, \lambda_i)$ denotes distance between an item and the centroid of ith cluster. K-means partitioning process can be summarized as follows:

1. Select K centroids randomly from the dataset.
2. Items are assigned to the cluster which centroid is closer to them.
3. Centroids are re-calculated to minimize the sum of distances to the centroid of all the items belonging to the same cluster.
4. Repeat steps 2. and 3. until no items change their cluster membership.

K-means algorithm is very efficient however the final clusters are very sensitive to the selection of the initial centroids and it has problems with outliers.

Association rule

Association rule mining algorithms try to find rules that predict the occurrence of an item based on the occurrences of other items in a transaction [10]. An association rule is an expression of the form $X \Rightarrow Y$ where X and Y are itemsets. The fraction of transactions that contain an itemset is its *support*. The frequency of an itemset is called *support count*. The *confidence* of the rule $X \Rightarrow Y$ is how often items in Y appear in transactions that contain X . The goal of association rule mining is to find all rules that have *support* $\geq$ *min support threshold* and *confidence* $\geq$ *min confidence threshold*.

Decision Trees

Based on the way tree graphs are bluilt, decission trees builds a predictive model that maps the input to a predicted value based on the input's attributes [11] . Each node of the tree corresponds to an attribute and the link from a parent to a child node corresponds to a possible value (or set of values) of the attribute. At each node an attribute is chosen as a split attribute. Then for each possible attribute value there is a link to a child node so that each child receives as inputs all items that have the appropriate value of the attribute that corresponds to the child-node. One of the attributes is pre-defined as the target attribute. The process repeated until all the items that goes to the node have the same target attributes value or until the number of items reaches a certain threshold. In collaborative systems attributes refer to the feedback provided by the user while in content-base approaches the attributes are the content features.

Bayesian Classifiers

These classifiers are based on the Bayes' theorem and the definition of conditional probability. The probability of a model given the data (posterior probability) is proportional to the product of its likelihood times its prior probability as stated in the Bayes' theorem. Given a set of N observations $\mathbf{X} = \{\mathbf{x_1}, ..., \mathbf{x_N}\}$ the goal is to find the class C_k that maximizes the posterior probability of the class given the data. Applying the Bayes' theorem:

$$P(C_k|\mathbf{X}) \propto P(\mathbf{X}|C_k)P(C_k) \tag{2.9}$$

A common Bayesian classifier is the *Naive Bayes Classifier*, that assumes the observations are independent to estimate the conditional probability $P(\mathbf{X}|C_k)$ so that:

$$P(\mathbf{X}|C_k) = P(\mathbf{x}_1|C_k)P(\mathbf{x}_2|C_k)...P(\mathbf{x}_N|C_k) \tag{2.10}$$

Naive Bayesian classifiers are robust to outliers in the observations data set

but the independence assumption can not be always applied.

Artificial Neural Network (ANN)

It is a structure of inter-connected nodes and weighted links. ANN assembly is inspired in the architecture of the biological brain, so nodes are called *neurons*. The network have the ability to learn a classification problem after being trained with sufficient data [10] .

The main advantage of ANNs is that, deppending on the activation function, they can estimate nonlinear functions and model complex data relationships. An important disadvantage is the great difficulty to find the ideal topology of the network given a problem and that it would act as a lower bound for the classification error [1] .

2.1.3. Advantages and drawbacks of CF systems

Collaborative-filtering systems have been widely used because of several advantages:

- Simplicity. They are relatively easy to understand and simple to implement when comparing to content-based systems that need a content characterization.
- Justifiability. Collaborative systems don't behave as a 'black box'. Predictions can be justified and understood.
- Efficiency. One of its main advantages is their computational efficiency, specially memory-based systems because they don't need any training or data content analysis.
- Serendipity. Unlike most content-based recommendations, collaborative filtering has the ability to recommend items that are different from the ones that the user have rated. The user is not always recommended items with the same characteristics.

On the other hand collaborative-filtering has some drawbacks:

- Cold start problem. When a new user registers to the system or a new item is added to the catalogue there is not enough data associated to those users or items. In the case of the new item it means that it will not be recommended until a number of users rate it while in the case of the new user it implies that the system is not able to compute a proper recommendation.
- Popularity bias. Items that have been rated by more users are more likely to be recommended to other users.
- Scalability. Typically computation grows linearly with the number of users and items so an algorithm that is efficient at a certain volume of items and users might not be able to maintain the quality of the recommendation process.
- Synonymity. In some databases very close items or even the same item, can have the different names or entries. They are treated as independent items worsen the systems efficiency and performance.

2.2. Content Based

Systems that implement a content-based recommendation analyze descriptions of items previously rated by a user (or other user information depending of the available data) and build a model of user interests based on the attributes of those items. Then for the recommendation the user model is matched with items obtaining a relevance estimation for each one.

In a general content-based information filtering system, as illustrated in Figure 2.2.1, three main components can be identified [2] :

1. Content Analyzer. The objective of this component is to represent the content of the items. Feature extraction techniques are used to analyze data items. The resulting representation is the input of the other components, the *profile learner* and the *filtering component.*

2. Profile Learner. The user preferences' model is built based on features of rated items. As in collaborative-filtering systems ratings or preferences of users can be obtained from explicit ratings or inferred from users' behaviour.

3. Filtering Component. The user profile is matched with the items feature representation to obtain a binary judgement or a relevance estimation. They are computed using some similarity metrics (e.g. as described in section 2.1) .

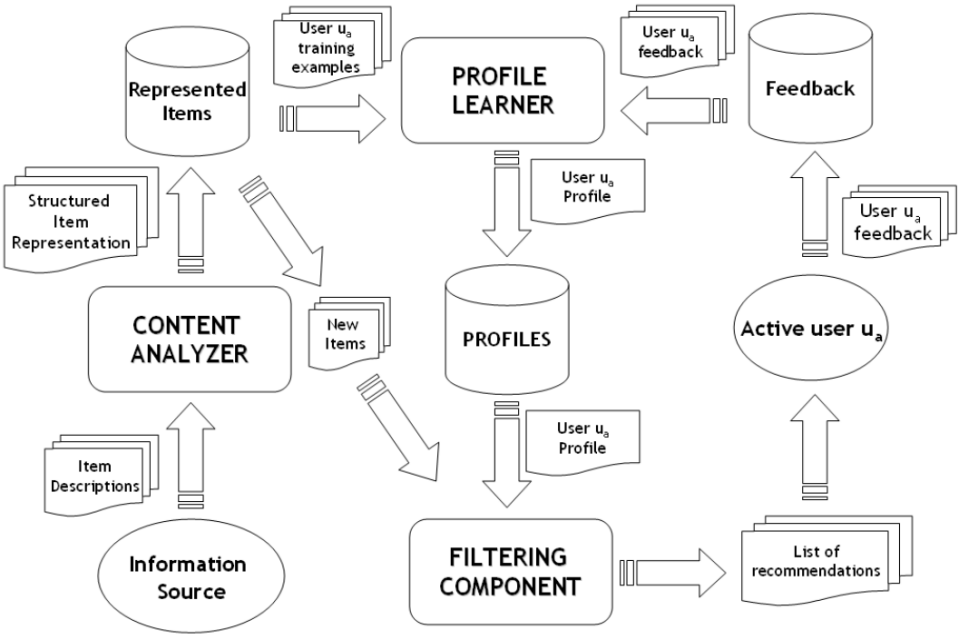


Fig. 2.3. General architecture of a CB recommender system. Source: [2]

The implementation of these scheme depends on the nature of the data on each recommendation domain. The case of content-based music recommenders is discussed in chapter 3.

2.2.1. Advantages and Drawbacks of CB systems

The content-based information filtering technique has several advantages compared to the collaborative one:

- User independence. Collaborative filtering approaches need other users ratings to find users with similar tastes, so the recommended items will be the most liked by the neighbours. A content based recommender do not take into account other users so a big community of users is not needed to provide a quality recommendation.
- Transparency. It is easier to understand why items were recommended or not analysing the descriptions or features of those items. In collaborative systems, in contrast, the only explanation is that other unknown users with similar tasted liked that item.
- New items. New items are equally likely to be recommended as the old ones. Collaborative methods need that a significant number of users rate a new item for it to be recommended as the older items.
- Popularity bias. As in the previous case items that have been rated by more users (more popular items) are more likely to be recommended than less rated items in a collaborative systems. That doesn't happen in a content-based system since it only takes into account features or descriptions of items.

But content-based methods have also some drawbacks that affect their performance in real applications:

- Serendipity problem. Content-based systems recommendations tend to have a limited degree of novelty since their characteristics are similar to the items already rated by the user. This could be managed by introducing in the system some randomness and in certain cases not recommending items that are too similar

- Description dependence. The effectiveness of the recommendation is directly dependent to the quality of the content description.
- New user problem (cold-start problem). When a new user registers to the system there is not enough information associated to that user so the system is not able to properly recommend items to the user.
- Computational complexity. Depending on the nature of the items, extracting descriptors and running the recommendation algorithm may be a time and memory consuming task.
- Scalability. If the number of items grows considerably the amount of resources needed to run the algorithm can make the system not feasible.

2.3. Hybrid Systems

A recommender system can also combine strategies from different approaches. In that case the system is classified as 'hybrid'. Hybrid systems commonly have better prediction accuracies than 'pure' systems separately because their objective is to exploit the advantages of different systems. Researcher R.Burke described several hybridization methos in [12] :

- Weighted. The final score of a recommended item in a weighted hybrid system is computed as a combination of the individual scores obtained by the different recommendation techniques that compose the system. An example could be a linear combination of the scores given by a collaborative and a content-based approach.
- Switching. Switching hybrid methods use some criterion to switch between recommendation techniques depending on the item. A good example is the DailyLearner system [13] that uses a content-based method in first place but if the obtained recommendation doesn't have enough level of confidence then the collaborative system is used.

- Mixed. Several recommendation techniques are used simultaneously and then recommendations from more than one system are presented to the user.
- Feature combination. In this type of hybridization the output of a collaborative system is used as an additional feature by a content-based system. This reduces considerably the cold-start problem of collaborative methods in the final recommendation.
- Cascade. One recommendation method is used first and then a second one, but focusing only on the items discriminated by the first system. Cascading methods are useful when the second recommendation technique requires a substantial data reduction.

2.4. Reducing Dimensionality

In recommender systems dimensionality reduction methods are very used because typically matrices are large and sparse (in the case of collaborative filtering) or to reduce dimensions of the item (or user) feature representation. One of the most relevant algorithms is *Principal Component Analysis (PCA)*.

Principal Component Analysis.

This algorithm is one of the oldest and best known techniques of multivariate analysis. PCA is a widely used technique for dimensionality reduction, lossy data compression, feature extraction and data visualization [14] . For dimensionality reduction, the idea is to reduce a large dataset that contains interralated variables without losing much of the original variability of the data. Original data is transformed to a new set of uncorrelated variables (the principal components) listed in order of contribution to the total variation of the original data [14].

Data points are projected onto a lower dimensionality space (the principal subspace) which its main characteristic is that the orthogonal projection of the original datapoints onto this subspace maximizes the variance of the projected points [15] .

Let's consider the projection onto a one-dimensional space ($M = 1$). If the original dimensionality of the problem is D, the direction of the new space can be defined using a vector $\mathbf{u_1}$. For convenience it is a unit vector so that $\mathbf{u}_1^T\mathbf{u}_1 = 1$ (we are interested in the direction of the vector, not its magnitude). Because $M = 1$ each data point $\mathbf{x}_n$ is projected to a scalar value $\mathbf{u}_1^T\mathbf{x}_n$. The mean point of the data projected in the subspace is $\mathbf{u}_1^T\overline{\mathbf{x}}$, where $\overline{\mathbf{x}}$ is the mean of the sample set composed of N observations, defined as

$$\overline{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n \tag{2.11}$$

The variance of the projected data is given by

$$\frac{1}{N}\sum_{n=1}^{N}\left(\mathbf{u}_1^T\mathbf{x}_n - \mathbf{u}_1^T\overline{\mathbf{x}}\right)^2 = \mathbf{u}_1^T\mathbf{\Sigma}\mathbf{u}_1 \tag{2.12}$$

where $\mathbf{\Sigma}$ is the covariance matrix defined as:

$$\mathbf{\Sigma} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \overline{\mathbf{x}})(\mathbf{x}_n - \overline{\mathbf{x}})^T \tag{2.13}$$

Now the goal is to maximize the variance of the projected data with respect to $\mathbf{u}_1$. The normalization condition $\mathbf{u}_1^T\mathbf{u}_1 = 1$ prevent $\|\mathbf{u}_1\| \to \infty$. so the maximization is constrained. To enforce this constrain a Lagrange multiplier λ_1 is introduced and then the following unconstrained maximization with respect to $\mathbf{u}_1$ is done:

$$\left(\mathbf{u}_1^T\mathbf{\Sigma}\mathbf{u}_1 + \lambda_1(1 - \mathbf{u}_1^T\mathbf{u}_1)\right)' = 0 \tag{2.14}$$

So we get

$$\mathbf{\Sigma}\mathbf{u}_1 = \lambda_1\mathbf{u}_1 \tag{2.15}$$

Left-multiplying both terms by $\mathbf{u}_1^T$ and simplifying $\mathbf{u}_1^T\mathbf{u}_1 = 1$ we get that the variance is given by

$$\mathbf{u}_1^T\mathbf{\Sigma}\mathbf{u}_1 = \lambda_1 \tag{2.16}$$

It means that the maximum variance will be obtained for $\mathbf{u}_1$ equals to the eigenvector with the largest eigenvalue (λ_1). That eigenvector is known as the first principal component.

Considesing now a general case of an M-dimensional projection space, the optimal linear projection for which the variance of the projected data is maximized is given by the M eigenvectors $\mathbf{u}_1, ..., \mathbf{u}_M$ of the data covariance matrix $\mathbf{\Sigma}$ that correspond to the M largest eigenvalues $\lambda_1, ..., \lambda_M$ [15] . The projection of a set of observations Y onto the new subspace can be calculated as $Y = XU$, where X is the matrix of N-dimensional original observations. The remaining data variability in percentage after the dimensionality reduction can be calculated as:

$$V_M = \frac{\sum_{i=1}^{M} \lambda_i}{\sum_{i=1}^{N} \lambda_i} \times 100 \tag{2.17}$$

Some useful properties of the final principal components are:

- Their variance is in decreasing order ($\lambda_1 > ... > \lambda_M$).
- They are uncorrelated, that is $cov(\mathbf{x}\mathbf{u}_i, \mathbf{x}\mathbf{u}_j) = \mathbf{u}_i^T\mathbf{\Sigma}\mathbf{u}_j = \lambda_j\mathbf{u}_i^T\mathbf{u}_j = 0$ for $i \neq j$, because the eigenvectors u are orthonormal so the matrix $\mathbf{U}$ of eigenvectors is orthogonal $\mathbf{U}\mathbf{U}^T = I$ where I is the identity matrix.

2.5. Evaluating Recommender Systems

There are several metrics that can be used to evaluate the quality othe a recommendation algorithm [4]. Depending on the filtering technique some metrics might be more suitable than others. Also some metrics quantify the ability of the system to find good items, like precision and recall metrics, while others measure the ability

to rank in the initial positions of the recommendation list good recommendations, like MAP, NDCG or MPR.

Mean absolute error (MAE).

This metric measures the quality of the recommender by computing the average absolute deviation between the estimated rating and the actual rating provided by the users, it is computed as:

$$MAE = \frac{1}{|T|} \sum_{r_{u,i} \in T} |r_{u,i} - \hat{r}_{u,i}| \tag{2.18}$$

where $r_{u,i}$ represents the real rating of user u to item i , $\hat{r}_{u,i}$ represents the estimated rating of user u to item i and T is the test set of users ratings.

Root mean square error (RMSE).

As in the MAE it is centered on the deviation between the actual rating and the estimated one but this metric squares the error, which penalizes larger deviations. It is calculated as:

$$RMSE = \sqrt{\frac{1}{|T|} \sum_{r_{u,i} \in T} (r_{u,i} - \hat{r}_{u,i})^2} \tag{2.19}$$

Precision.

This metric was designed for binary relevance judgements and it is defined as the ratio of recommended items that are relevant to the user in the test set T over the total number of recommended items. For the top K recommended items the precision is defined as:

$$P_u@K = \frac{|L_u \cap \hat{L}_u|}{|\hat{L}_u|} \tag{2.20}$$

where L_u denotes the set of relevant items for user u in the test set T and $\hat{L}_u$ represents the recommended set with the K top rated items according to the prediction for user u. The global *precision* is calculated averaging the individual $P_u@K$ of all the users in the test set.

Mean average precision

It is defined as the average of the overall precision value $P@K$ for different lengths of the recommendation set K as follows:

$$AP@K = \frac{1}{N} \sum_{i=1}^{K} rel(i) P@i \tag{2.21}$$

where $rel(i)$ is equal to 1 if the ith recommended item is relevant or 0 if it is not, N is the total number of relevant items.

Recall

is the fraction of relevant items that are in the recommendation list over the total number of relevant items. For the top K recommended items recall is defined as:

$$R_u@K = \frac{|\mathbf{L}_u \cap \hat{\mathbf{L}}_u|}{|\mathbf{L}_u|} \tag{2.22}$$

The overall relevance of the system is computed as the average of the recalls obtained for the different users.

Normalized discounted cumulative gain (NDCG)

is a metric that is widely used for evaluating the effectiveness of information retrieval systems. In the context of recommender systems it measures the ranking quality of the recommendations. The *discounted cumulative gain* for a user u

DCG_u is defined as:

$$DCG_u = \sum_{i=1}^{N} \frac{r_{u,i}}{log_2(i+1)} \tag{2.23}$$

where $r_{u,i}$ is the actual rating provided by the user u for the item ranked at position i in the recommendation list with length N. Due to the different criteria when rating items, DCG_u values are not directly comparable between users. The $NDCG$ is used instead, and it is computed from the DCG_u and the ideal DCG of the user $IDCG_u$ as:

$$NDCG_u = \frac{DCG_u}{IDCG_u} \tag{2.24}$$

The overall $NDCG$ of the system is computed by averaging the individual $NDCG_u$.

3. Music Representation and Recommendation

3.1. Introduction

The reaserch field of Music Information Retrieval (MIR) is relatively recent and has grown significantly in less that two decades as the result of a range of factors such us the increase of personal computers' computing power, the improvement of audio compresion techniques and more recently the raise of music streaming services popularity [3] [16]. Some subfields of the research area are [3]:

- *Feature extraction* such as beat tracking, melody extraction and timbre description.
- *Music similarity measurement*
- *Music classification.* Examples of classifications are genre classification, mood recognition and instrument classification.
- Applications such as *music recommendation*, *context-aware and adaptive systems*, *playlist continuation* and *audio fingerprinting*.

This chapter is focused on two important MIR tasks: music feature extraction and music representation.

3.2. Music Representation

A large list of audio descriptors have been designed for different purposes. According to the definition given in the MPEG-7 standard [17], a content descriptor

is a ‘distinctive characteristic of the data which signifies something to somebody’. This standard was released in 1999 and was designed to be a standardization of the way multimedia content is described. It only standardize the way the descriptions are structured, not how they are obtained or used.

One important aspect of music content description, as other systems dealing with audiovisual content,is that it has the drawback of the *semanticgap*. That is, as defined by Smeulders et al. in [18], ‘ the lack of coincidence between the information that one can extract from the (sensory) data and the interpretation that the same data has for a user in a given situation’.

Also that characterization should be useful for a specific purpose. It might not be needed the same information in a musical instrument discrimination task than in a music recommendation one. One way to do this characterization in terms of human perception is manually associate the tracks with some tags that define them, but it is not possible when working with big databases. It is then more convenient to compute those characteristics automatically and as fast as possible.

3.2.1. Features Taxonomy

Gouyon et al. [19] and Leman et al. [20] among others suggested three different criteria to classify music content descriptors: *level of abstraction* (low,mid and high-level), *temporal scope* (instantaneous, frame-based or global scope) and according to the different *musical facets* (harmony/tonality, melody, rhythm, timbre/instrumentation, editorial, textual, bibliography).

1. **Temporal scope**. The description provided by a feature may apply to the whole signal (e.g. the attack duration of a sound), that is the case of *global descriptors* . Also descriptors can be computed for each time frame, then they are *instantaneous descriptors* (e.g. the spectral centroid of a signal, which can vary along time). In order to compute the final features the temporal vectors of the instantaneous descriptors are processed by a modeling module. [21]

2. **Levels of abstraction**. Three levels of abstraction are considered to classify music content description:

    - Low-level descriptors. Computed directly from the signal or from a derived representation, (e.g. from the Fourier Transform of the signal). They are suitable for computer systems but they are not understandable for most of the users unless they have any technical background.
    - Mid-level descriptors. They characterize aspects of the music that are more 'objective' (e.g. like chords, keys or timbre descriptors)but are not technical aspects of the signal. Mid-level descriptors are understandable for users with a general knowledge about music theory.
    - High-level (or *semantic*)descriptors. These descriptors are computed from Mid and Low-level descriptors. They are designed to characterize music audio with terms that humans use to describe music in a more subjective way (e.g. western or non-western music, genre classification).

3. **Facets of Music Information**. Researcher J.S. Downie proposed to analyze music information according to seven facets [16]: pitch, temporal, harmonic, timbre, editorial, textual and bibliographic facets. Those facets are not mutually exclusive, since some concepts can be classified in several facets depending on the context. They can also depend on other facets like the harmonic one, that depends on the pitch and temporal facets. Descriptors organized according to this classification are of a high-level of abstraction and most of them are computed from lower-level descriptors (e.g. timbre descriptors are computed from spectrum features).

    - *Pitch* facet. Information about the tones present in the music. Pitch is a function of its fundamental frequency. There are different representations of the pitch, like a symbolic representation as notes written on a stave or as numbers on a tablature for example.
    - *Temporal* facet. It covers everything related to the duration of musical events.

- *Harmonic* facet. The music *harmony* as the units (formed by a set of tones) that we analyse by hearing. Typically harmonies are constructed by chords, but sometimes they can be present in an interval (two different tones together). The concept is quite abstract and is linked to the western music tradition so the description of non-western music in terms of its harmonic facet is difficult. Another difficulty is the lack of consensus in the classical music community about some harmonic analysis.
- *Timbre* facet. It includes the information related to the tune color. It is what help us differentiate the same tune played by a trumpet or a mandolin. One instrument may have different colors depending on the way it is played.
- *Editorial* facet. They are mainly performance instructions. From dynamic instructions (e.g $crescendo$, f, mp) fingering, articulation, bowing to any other symbol related to performance. One important difficulty of this facet is that the information can be textual or iconic. Another difficulty is that there are symbols very specific for an instrument and it could be the case that one symbol in different music sheets can have different meanings. Specially in contemporary compositions there might be symbols invented by the author.
- *Textual* facet. Operas' libretti, lyrics of songs or any other music composition are included here. There are certain lyrics or sentences that have a melody associated to them, but since there are translated versions of a lot of songs a melody may have several texts associated to it.
- *Bibliographic* facet. It includes information about the title, composers, performers, catalogue number, etc. It is music meta data rather than information about the content of the audio.

3.2.2. Feature Extraction

Time and Frequency Representation

Music features are computed from the time and frequency representations of the signal [3] . The time-domain representation is used to obtain descriptors related to the temporal evolution of the waveform, like the *zero-crossing rate* that measures the number of times the signal changes from positive to negative sign and vice versa. In Figure 3.1 there are two examples of the time-domain representation of two tracks from very different styles:

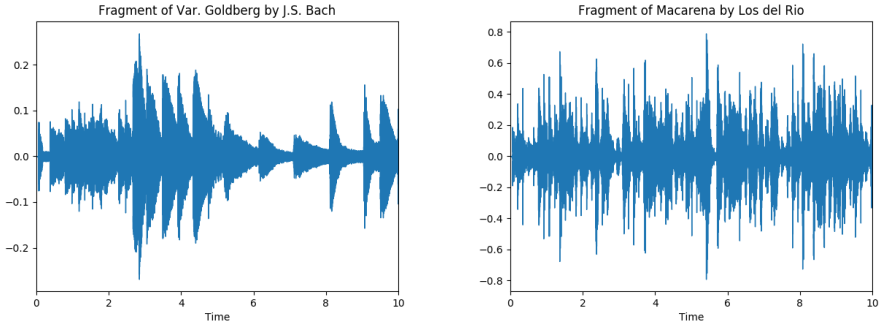


Fig. 3.1. Time representation of 'Chaconna' by J.S. Bach and 'Macarena' by Los del Rio

The *frequency spectrum* of a time-domain signal is its representation in de frequency domain. In speech recognition using the Fourier Transform (FT) is very common and since several of the music signal analysis techniques were based on the speech recognition pre-existing ones, the FT is also very important in MIR. When working with digital audio the discrete version of the FT is the Discrete Fourier Transform. One of the most used techniques for spectrum analysis is the Fast Fourier Transform (FFT) because it is an efficient algorithm for calculating the DFT [22]. The spectral content of an audio signal changes very quickly so sometimes it is more useful to compute the spectral representation over short time segments of the signal using the Short-Time Fourier Transform (STFT). The STFT is computed windowing the time-domain signal with an sliding window of the desired type (e.g. rectangular, Hamming, triangular) and the chosen overlap

between consecutive windows, and then applying the FFT to the individual signal segments. In Figure 3.2 the final result of applying the STFT is illustrated as a power spectrogram. The main parameters that influence the analysis are the window size, the overlap and the type of window used in the analysis.

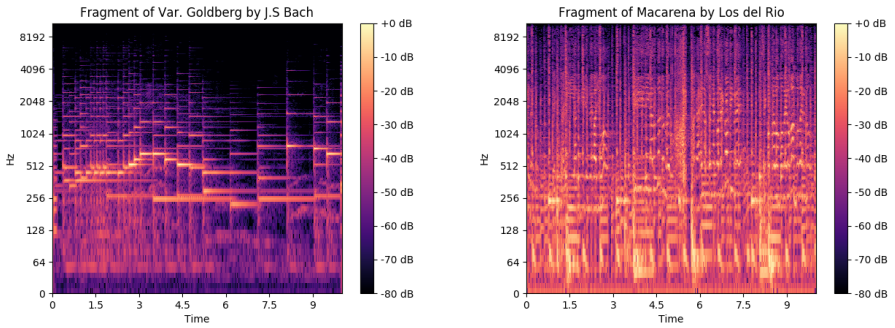


Fig. 3.2. Soectral representation of 'Chaconna' by J.S. Bach and 'Macarena' by Los del Rio

Low level descriptors extraction

The list of low level features is considerably large because for the different MIR tasks some specific features were designed. For the CUIDADO project [21] a large set of audio features was described. Most low-level features describe loudness and timbre and are extracted from the time representation and frequency representation of the audio signal. In Figure 3.3 the general extraction process of low-level descriptors is depicted.

Frequency Cepstrum Coefficients

A powerful spectral feature widely used in speech recognition and in MIR are the Mel Frequency Cepstrum Coefficients (MFCCs), because they represent in a compact way (as a finite number of coefficients, typically 13 in the literature) the signal spectrum. They were proposed for speech recognition and later B. Logan [23] demonstrated the same process could be applied for music modelling.

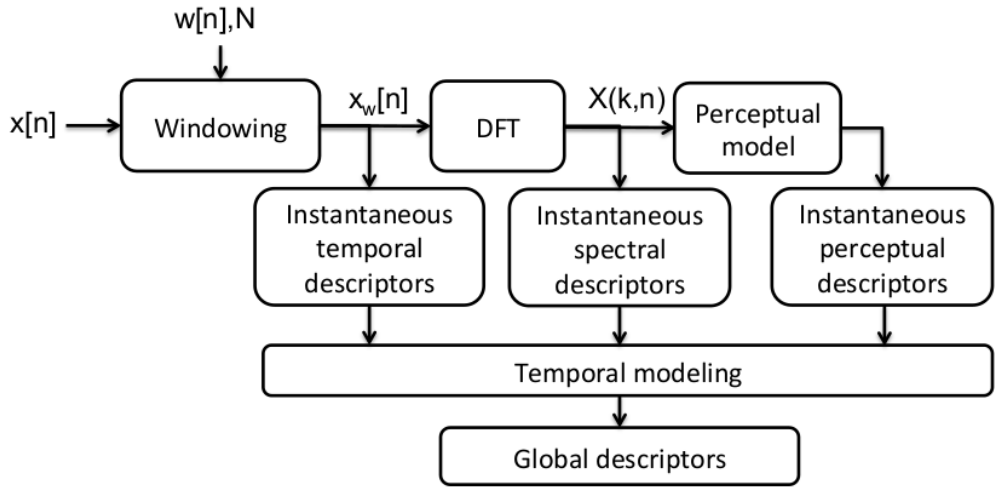


Fig. 3.3. Diagram of low-level features extraction. Source: [3]

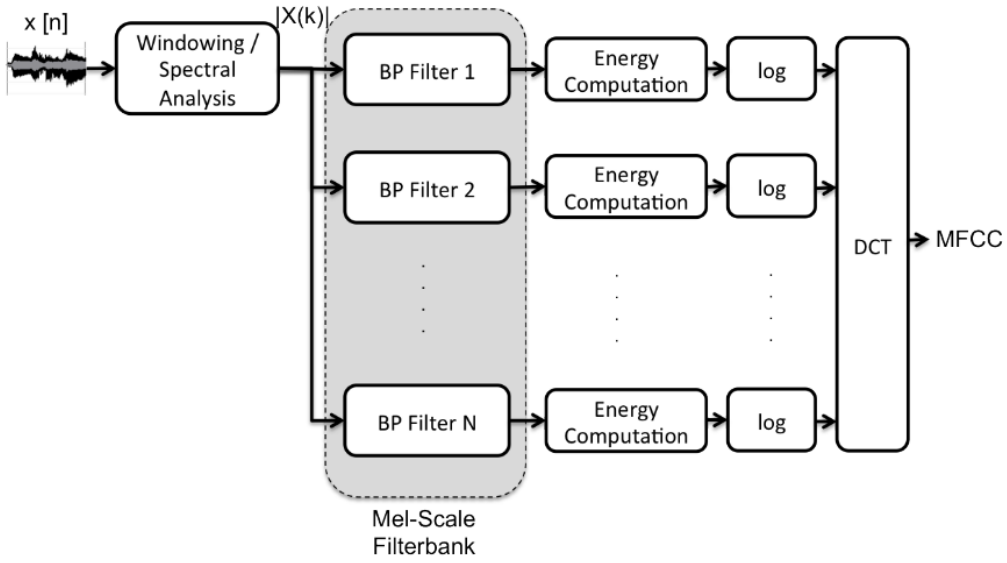


Fig. 3.4. Block diagram of the MFCC extraction. Source: [3]

The main steps are illustrated in Figure 3.4. The typical windowing function is a Hamming window to remove edge effects. The spectral representation is filtered

using a Mel-scale filterbank[3]. The DFT (or FFT) of each frame is computed and then the logarithm of the amplitud spectrum is obtained. This step is based on human sound perception, because it has been shown to be approximately logarithmic in terms of loudness perception and that it prioritizes the amplitude of the spectrum over the phase information. Then the log spectrum amplitud is converted to Mel-frequency scale to emulate the human spectral perception. This is done filtering the log-amplitude spectrum with a filterbank of overlaping triangular windows which bandwidths follow the Mel-frequency scale. Finally the DCT is applied to obtain decorrelated coefficients for each frame. The low order MFCCs gives information about the spectral envelope and the higher order ones describe the fast variations of the spectrum. MFCCs are commonly used in speach recognition but also in MIR tasks such as musical instrument identification [25], music similarity [26] and music recommendation [27]

In the Annex an extensive list of low-level music descriptors is provided. The list corresponds to low-level features that can be extracted using the audio library Essentia [4].

3.3. Music Recommendation

Music recommendation is an inportan MIR tasks and it has been approached in many different ways. Music similarity between tracks is a very close MIR task that can be used to compute recommendations, however it is also useful for other issues such us song cover identification.

Data about music is essential for a music recommender. Research studies on music recommendation uses three different sources of information [28]: manual expert annotations, automatically extracted annotations from the Internet (e. g. keywords from web pages and social media or social tags) and collaborative data generated by users (such us songs or artists ratings or playcounts).

[3]Mel-Scale is a perceptual scale that tries to measure the psychological perceived pitch of the soud [24]

[4]`http://essentia.upf.edu`

Collaborative filtering methods can also be used for music recommendation as they are directly applicable to most of the recommendation domains. The music recommender system proposed in [29] was one of the first that applied a collaborative filtering method to music recommendation.

Appart from collaborative filtering, may approaches are focused on information extracted from the music audio. D. Bogdanov in [28] classified many music recommenders according to the content information they used:

- *Timbral* information. Typically represented by the MFCC.
- *Temporal* information, such as loudness evolution with time, rhythm information or structure of the music.
- *Tonal* information, related with harmonies, chords and key.
- *Inferred semantic* information, such us extensive automatically extracted genre tags or unsupervised clustering methods in the feature space.

Once the music information is extracted, different techniques can be used to compute the recommendations. Some that have been explored for music recommendation are [28]:

- *Distance based ranking*, using similarity metrics such as euclidean distance, cosine distance and Pearson correlation.
- *Discriminative models*, such us kNN algorithm or support vector machines (SVMs).
- Probabilistic generative models, e.g. GMMs or hidden Markov models (HMM).
- *Automated reasoning* using ontologies.

In practice, many of the music similarity measures are based on timbral similarity with MFCCs (e.g. [27],[30], [31], [32]) in a similar way as they are used

for speech recognition [33]. There are different approaches that use GMM and MFCC, e.g. MFCCs and GMM with initial parameters obtained using k-means algorithm [32], the same approach but obtaining init parameters using a Universal Backgruond Model (UBM) and a GMM and using a GMM supervector [31]. GMMs have also shown to be performe well when using other features e.g. semantic descriptors [27]

4. Description of the Proposed Systems

4.1. Introduction

In this chapter the database used, the feature extraction process and two different systems are described. After evaluating several approaches of recommender systems, as detailed in chapter 5, two systems were chosen. Each one of them is more convenient for a different situation. The first system is purely content-based and uses the Pearson correlation between the feature representation of all tracks to rank the items according to their minimum distance to any track of the user's playlist. It is the system that obtained better results but it also was the slowest one. This system is proposed for situations with enough computational power to store all correlations between items so that it is not necessary to to compute all distances each time a recommendation is requested or for situations were instant recommendations are not required. The second proposed system is an hybrid system that filters the database using kNN algorithm and models the playlist of the user using GMM. This second system is considerably faster than the first one but its results were not as good, so the better architecture depends on the

4.2. Description of the Database

Two potentialy usefull databases were found. One is the *Million Song Dataset (MSD)* [34] , originally a collaborative project between The Echo Nest [5] and LabRosa [6]. It is a freely-available dataset of audio features and metadata of a million popular music tracks. The MSD also contains complementary datasets, one of which is called the *Tase Profile* subset. This complementary dataset provides real user play-counts. The audio analysis and the corresponding catalogue

[5] http://the.echonest.com

[6] https://labrosa.ee.columbia.edu/

ID of the songs were provided through The Echo Nest API in the past but in June 2016 Echo Nest shut down their API. This is a problem because the MSD uses Echo Nest identifiers to refer each track and metadata (including names of artists, songs and tracks), and they were only accessible through the API. Inside the *Acoustic Brainz* project [7] they provide mappings of Echo Nest song identifiers in the MSD to IDs of other services if they are available (e.g. identifiers of Spotify, 7digital and musicbrainz catalogues) [8] . The MSD can still be used however this year another dataset was released and it has some characteristics that make it more interesting. Spotify released a database called *The Million Playlist Dataset* (MPD) [9] that contains 1 million playlists created by users of the Spotify platform. An important characteristic of playlists is that they are intended to be listened in sessions. Songs that are added to a playlist typically share some characteristics that make the user want to listen them one after the other. Additionally metadata of all tracks can be obtained through the Spotify API directly, since in the MPD they provide Spotify identifiers. Also a 30 seconds preview is available for most of the tracks.

The Million Playlist Dataset

In the context of the RecSys Challenge 2018 Spotify released a database called *The Million Playlist Dataset* (MPD) that contains 1 million playlists created by users of the Spotify platform in the United States. Those playlists were created between January 2010 and October 2017 and selected randomly from the huge amount of playlist created by users that fulfil the selection criteria:

- Creator is at least 13 years old and lives in the Unated States.
- Public at the time MPD was generated
- Contains at list 5 tracks

[7] http://acousticbrainz.org

[8] https://labs.acousticbrainz.org/million-song-dataset-echonest-archive/

[9] *Million Playlist Dataset*, official website hosted at https://recsys-challenge.spotify.com/

- Contains no more than 250 tracks
- Contains at least 3 unique artists
- Contains at least 2 unique albums
- Has no local tracks (local tracks are non-Spotify tracks that a user has on their local device)
- Has at least one follower (not including the creator)
- Was created after January 1, 2010 and before December 1, 2017
- Does not have an offensive title
- Does not have an adult-oriented title if the playlist was created by a user under 18 years of age

Playlists are stored in groups of 1.000 playlists in JSON files. Playlists contain a set of tracks and additional metadata that gives information about: name of the playlist, user-provided playlist description, instant of last update, number of tracks, artists, albums, followers and editing sessions, total duration of all tracks of the playlist and whether the playlist is collaborative or not. Every track consists in: track name and URI, artist name and URI, album name and URI, position in the playlist and duration. Using the provided track URI a preview of 30 seconds can be obtained using the Spotify API, but it is not available for all tracks.

Final dataset

The MPD was pre-processed for the purpose of this project to reduce the computational cost of each experiment and reduce the difficulty of the collaborative filtering. The final set has 3,142 playlists with the following characteristics: they have at least 20 tracks per playlist and eack track of the playlist appears in at least 20 playlists of the final set. Album and artist information is not taken into account for the recommendation so we only filter the dataset by track related characteristics. In the dataset there are 6.034 unique tracks, all of them with available preview

though the Spotify API. The number of playlists were not increased because the time of execution in some of the evaluated systems in chapter 5 is directly dependent on the database size, so to be consistent all recommenders were tested using this reduced database.

4.3. Recommendation

4.3.1. Feature Extraction

The feature extraction process was based on [28] and has three steps, as depicted in figure 4.1: in the first one low-level descriptors are extracted from the audio file, then a high-level representation is obtained using the low-level descriptors and at the end a lower dimension representation of the feature vector is obtained using PCA. At the end of the process we have a semantic representation of the track that will be used by the content-based algorithm.

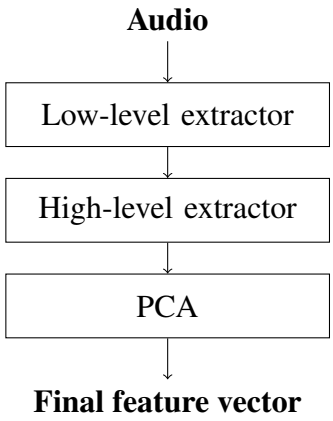


Fig. 4.1. Feature extraction process.

Feature extractors from the audio library Essentia [35] are used to obtain the low-level and high-level representations. Essentia is an open-source library for audio analysis and audio-based music information retrieval created and supported by the Music Technology Group of the Universidad Pompeu Fabra. This extrac-

tors are suited for batch processing on large music sets so the process is quite fast considering the amount of features extracted.

4.3.2. Low level descriptors

The set of low-level features describes a wide range of aspects (or facets) of the music, so the number of low-level descriptors is large. Once the low-level features are computed they are used to obtain the the high-level representation of the track in the next step.

Before computing low-level descriptors the audio signal is re-sampled to 44kHz, converted to mono and normalized using its ReplayGain value (which is a gain value for a given track for loudness normalization according to the ReplayGain specification) by the Essentia low-level extractor. This is done because otherwise comparisons between tracks with different characteristics of the digital audio signal could lead to erroneous results. Frame-wise descriptors are summarized by their statistical distribution.

In Essentia low-level descriptors are divided in three groups: low descriptors, tonal descriptors and rhythm descriptors. Frame-wise descriptors are summarized by their statistical distribution. Below is an overview of the different groups of low-level descriptors, a detailed description is given in Annex I.

Low-level descriptors

In this group there are spectral and time-domain descriptors. The spectral content of the signal is analysed in different perceptual scales: Mel, Bark and Equivalent Rectangular Bandwidth (ERB). Some features describe the spectral distribution, e.g. central moments, crest and flatness that are computed over energies in Bark bands, Mel bands and ERB bands. Other features that are used to describe the spectral shape are the spectral centroid, spectral spread, spectral skewness and spectral roll-off. Other features that belong to this group are the average loudness, which gives information about the dynamic range of the signal, the silence rate for several thresholds and the zero-crossing rate, which gives information about

the noisiness of the signal.

Tonal descriptors.

In this category we have the general key of the track, information about chords like the chord strength, the sequence of chords or the chords change rate. Also some high-resolution chroma features are computed. They give information about the tuning system or scale which is very useful for comparative analysis of music from a western or non-western traditions.

Rhythm descriptors.

Features of this group are the onset rate, danceability (which is a musical descriptor proposed by S. Streich and P. Herrera in [36]) and some descriptors characterizing the BPM histogram and the actual beats (their count, positions in time, loudness and band ratio).

High level (semantic) descriptors

High-level features try to reduce the semantic gap so that they can describe music content closer to human perception than the low-level descriptors. They are computed using several classifiers, each of them trained on a different semantic dimension such as genre, moods, instrumentation, rhythm and tempo [28] . Classifiers were built using 20 ground truth music collections corresponding to 20 classification tasks. Support Vector Machines (SVMs) were used for the classification. Parameters of each SVM were found by a grid search with 5-fold cross-validation. Then classifiers were trained using the previously mentioned music collections, which low-level features were selected by the Correlation-based Feature Selection (CFS) method as described in [37] according to the specific music collection.

In Table 4.1 we can see that the set of features are focused on different characteristics of the music (e.g. genre, mood, instrumentation). Some models describe

Classifier	Category	Classes
G1	Genre	Alternative, blues, electronic, folk/country, funk/soul/rnb, jazz, pop, rap/hiphop, rock
G2	Genre	Classical, dance, hip-hop, jazz, pop, rhythm 'n blues, rock, speech
G3	Genre	Blues, classical, country, disco, hip-hop, jazz, metal, pop, reggae, rock
GEL	Genre	Ambient, drum 'n bass, house, techno, trance
CUL	Cultural	Western, non-western
MHA	Mood	Happy, non-happy
MSA	Mood	Sad, non-sad
MAG	Mood	Aggressive, non-aggressive
MRE	Mood	Relaxed, non-relaxed
MAC	Instrumentation	Acoustic, non-acoustic
MEL	Instrumentation	Electronic, non-electronic
MCL	Mood	5 mood clusters (more information in Annex I)
RPS	Tempo (perceptual)	Slow, medium, fast
RBL	Genre	Chachacha, jive, quickstep, rumba, samba, tango, viennese waltz, waltz
ODA	Cultural	Danceable, non-danceable
OPA	Cultural	Party, non-party
OVI	Instrumentation	Voice, instrumental
OTN	Tonal	Tonal, atonal
OTB	Timbre	Bright, dark
OGD	Voice gender	Male, female

Table 4.1. High-level descriptors models.

the track according to different music genre classifications. Although it could be case in which one track did not belong to any of the genres of the classification provided by one model, that information is still descriptive of the content of the

track. For example RBL classifies a track according to several music genres that correspond to a dance style. If one track does not belong to any of them but it is classified as waltz, that information could mean that the track is not rhythmic or energetic. So no classifiers are discarded at these point to compute the content description.

4.3.3. Final feature vector

Once all semantic descriptors are extracted, dimensions of the feature vectors are reduced using PCA. Dimensionality is reduced for several reasons:

- Playlists can have a minimum length of 20 tracks. The method used for building the GMMs from the library scikit-learn restricts the number of samples, so they must be greater or equal to the number of features.
- Descriptors taken from the same classifier might be highly correlated since they sum up to one. There are several multi class classifiers and using all classes may not give really unique and uncorrelated information.
- Some genre classifiers have similar or even the same classes. As in the previous case those features could be highly correlated.
- Reducing dimensions of track representation lightens the system so it takes less time to compute each recommendation.

The chosen number of components is 15 because, after computing the total explained variance as a function of the number of components as detailed in section 5.3 , the total explained variance for 15 components were more than 95%.

4.4. Proposed Content-based System

This is the system that obtained the best metrics of all the evaluated recommendation techniques and is also one of the more simple ones. It is a distance-based

method that recommends the tracks from the database that are closer to any of the user playlist tracks in the feature space.

First all distances between the tracks from the user playlist and database tracks need to be calculated. Distance is measured with the Euclidean distance of the feature representation of the tracks. Then tracks are ranked according to their minimum distance to any track of the user playlist.

4.5. Proposed Hybrid System

Content-based methods obtained better results than the collaborative ones but they also were considerably slower as can be shown in chapter 5. While the maximum average time per recommendation of a collaborative method was 11 seconds the fastest average time per recommendation of a content-based method was 128 seconds. Combining GMMs with kNN algorithm improved metrics and average time. Also in a human evaluation its recommendations were rated positively so also it was not the recommender system that obtained better results

The proposed hybridization method is cascade, as depicted in the diagram of the system in figure 4.2 . The kNN method is used as a first stage of the hybrid system to filter the database and provide a smaller set of tracks so that the final recommendation list provided by the content-based technique is computed faster.

4.5.1. Collaborative Filtering: kNN

After comparing kNN filtering and threshold filtering with different parameters, the variant that obtained better results was kNN with 30 neighbours, so it is the collaborative method used in combination with a content-based filtering method. Similarity between playlists is measured using the Pearson correlation coefficient because it is a widely used similarity measure in collaborative filtering that has shown good recommendation accuracies compared to other distance metrics [9]. The singularity of kNN method is that, after computing correlations between playlists, only the set of k playlists with higher correlation values are used to

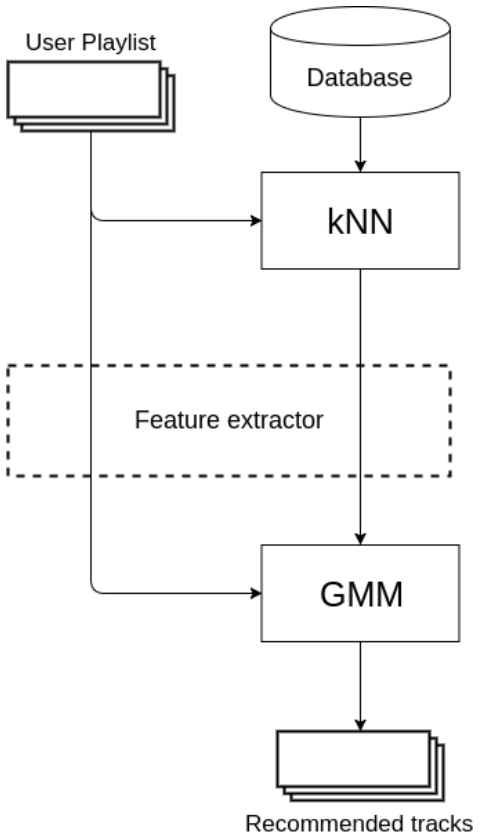


Fig. 4.2. Block diagram of the proposed hybrid recommender. The number of neighbours used in the kNN algorithm is 30 and the number of components in the GMM is 9.

compute the recommendation. The final score of each track is the sum of the correlation values of all playlists that contain that track divided by k, the number of neighbours. Tracks of the database are ranked in decreasing order of the score values and the a list with the first 1000 tracks are used by the content based method to obtain the final recommendation. Different lengths of that list were tested as explained in chapter 5 and 1000 was found to the be the smallest number that did not worsen the metrics.

4.5.2. Content-Based Algorithm: Gaussian Mixture Models(GMM)

In the content-based method first preferences of the user are modelled with a GMM. Then the probability of all the tracks of the collaborative filtering list is obtained and finally tracks are ranked in descendent order of probability.

It was one of the two the content based techniques that, in cascade with kNN, showed better overall performance. Its accuracies were the second best ones but its computing speed was significantly better than the speed of the system with better accuracies.

Music recommendation using GMM

Gaussian Mixture Models are probability generative models that can be used to estimate a probability density distribution [38]. A GMM can be seen as a linear combination of uni modal Gaussian densities, also called components. The model is defined by a set of parameters $\mathbf{\Theta} = (\mathbf{\Theta_1}, ..., \mathbf{\Theta_C})$ where C is the number of components of the model. Each component c_i is described by a set of parameters $\mathbf{\Theta_i} = \{\boldsymbol{\mu_i}, \mathbf{\Sigma_i}, \lambda_i\}$, which are the mean $\boldsymbol{\mu_i}$ the variance $\mathbf{\Sigma_i}$ and the weight of the components (the prior probability of the component) $\lambda_i = P(c_i|\mathbf{\Theta})$.

For a GMM with C components, that is described by parameters $\mathbf{\Theta}$ to model the user playlist probability distribution, a single track $\mathbf{x}$ would have a probability defined as the marginalization over all components of the model:

$$\begin{aligned} p(\mathbf{x}|\mathbf{\Theta}) &= \sum_{i=1}^{C} P(c_i|\mathbf{\Theta}) p(\mathbf{x}|c_i, \mathbf{\Theta}) \\ &= \sum_{i=1}^{C} P(c_i|\mathbf{\Theta}) \frac{1}{(2\pi)^{n/2}|\mathbf{\Sigma_i}|^{1/2}} \exp\Big(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu_i})^T \mathbf{\Sigma_i}^{-1} (\mathbf{x} - \boldsymbol{\mu_i})\Big) \end{aligned} \tag{4.1}$$

where n is the length of the feature representation of each track and the covariance matrix $\mathbf{\Sigma_i}$ is diagonal because feature vectors are transformed by PCA, so they are uncorrelated. The goal is to estimate parameters $\mathbf{\Theta}$ so that the GMM

models the distribution of the user playlist. For a user playlist $\mathbf{V} = \mathbf{v}_1, ..., \mathbf{v}_M$ where M is the length of the playlist and $\mathbf{v}_i$ is the n-dimensional feature representation of a track from the playlist, the maximum likelihood estimation of the model parameters would be defined as follows:

$$\begin{aligned} \boldsymbol{\Theta}_{ML} &= \text{argmax}_{\boldsymbol{\Theta}} \prod_{\mathbf{v} \in \mathbf{V}} P(\mathbf{v}|\boldsymbol{\Theta}) \\ &= \prod_{\mathbf{v} \in \mathbf{V}} \sum_{i=1}^{C} P(c_i|\boldsymbol{\Theta}) \frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma_i}|^{1/2}} \exp(-\frac{1}{2}(\mathbf{v} - \boldsymbol{\mu_i})^T \boldsymbol{\Sigma_i}^{-1}(\mathbf{v} - \boldsymbol{\mu_i})) \end{aligned} \tag{4.2}$$

Expectation Maximization algorithm is used to find the model parameters because Eq.4.2 is hard to solve analytically [38].

Expectation Maximization (EM) algorithm

The Expectation Maximization is an algorithm that iterates between estimating a posteriori class probabilies for each sample given some model settings (E-step) and re-estimating parameters of each component. Each EM iteration consists of the following steps [38]:

- **E-step:** Estimate hidden sample to component assignments h_{ij} for each sample $\mathbf{v_j}$ and component $\mathbf{c_i}$:

$$h_{ij} = P(c_i|\mathbf{v}_j) = \frac{p(\mathbf{v}_j|c_i)P(c_i)}{\sum_{c=1}^{C} p(\mathbf{v_j}|c_c P(c_c)} \tag{4.3}$$

- **M-step:** compute new component's parameters to maximize the joint distribution of component assignments and samples $p(\mathbf{V}, \mathbf{H}|\boldsymbol{\Theta})$, where $\mathbf{H}$ is the matrix with all sample assignments h_{ij}. For each parameter it means:

$$\boldsymbol{\mu_i^{new}} = \frac{\sum_j h_{ij} \mathbf{v_j}}{\sum_j h_{ij}}, \tag{4.4}$$

$$\boldsymbol{\Sigma_i^{new}} = \frac{\sum_j h_{ij} (\mathbf{v_j} - \boldsymbol{\mu_i^{new}})(\mathbf{v_j} - \boldsymbol{\mu_i^{new}})^T}{\sum_j h_{ij}}, \tag{4.5}$$

$$P(c_i)^{new} = \frac{1}{N} \sum_j h_{ij}. \tag{4.6}$$

Before EM, initial parameters are computed by k-means method as explained in section 2.1.2.1.Then this iterative process is repeated until a convergence threshold is reached.

Number of components estimation

After performing a grid search the number of mixtures that showed to model better users playlist for the recommendation was 9.

5. Experiments

In order to choose the architecture of the system that better suited the available databases and computing power, several variants of different algorithms were evaluated empirically. The experimental study was structured around four fundamental issues:

1. Recommendation approach: collaborative filleting, content-based filtering or hybrid system.
2. Collaborative technique: neighbour selection (threshold vs kNN).
3. Content based technique: distance or GMM.
4. Hybridization method: cascade or combination.

5.1. Procedure and evaluation

Each variant was tested using scripts in Python created for this thesis. Python was the chosen programming language because there are several Python libraries that are very useful for signal processing, audio analysis and machine learning tasks. The library Scikit-learn was used to model the GMMs and compute the PCA, distances were computed using the Scipy library and audio descriptors were obtained by the Essentia extractors [35] .

The evaluation metrics that were used to compare all recommendation techniques were the ones used in the RecSys Challenge 2018 because they are more appropriate than other metrics to evaluate this type of recommender (music recommender with no ratings). These are variants of the ones explained in section 2.5. Here we specify their exact implementation within the Challenge as provided by the organizers in an executable script. Although in the Challenge artist level

is also taken into account, for the purpose of this thesis only track level was evaluated. For an ordered list of recommended tracks R and the ground truth set of tracks G metrics are defined as:

- *Precision.* This metric is the ratio of the number of relevant items over the number of known relevant items:

$$Precission = \frac{|G \cap R_{1:|G|}|}{|G|} \tag{5.1}$$

  where $|G|$ denotes the size of the ground truth set and $R_{1:|G|}$ denotes the first $|G|$ items in the recommendation list. As it is an intersection of two sets of items it does not depend on the order of the recommended tracks within $R_{1:|G|}$.

- *Normalized discounted cumulative gain (NDCG).* As mentioned in section 2.5 *DCG* measures the ranking quality of the recommendation. It is an adapted version for situations were there no ratings are estimated. It takes higher values when relevant items are in higher positions of the recommendation list. The ideal *DCG* (*IDCG*) is the *DCG* when the recommended items are perfectly ranked. Then the *NDCG* is the ratio of both values. *DCG* and *IDCG* are calculated as follows:

$$DCG = \sum_{i=1}^{|R|} \frac{rel_i}{\log_2(i+1)} \tag{5.2}$$

$$IDCG = \sum_{i=1}^{|G|} \frac{1}{\log_2(i+1)} \tag{5.3}$$

  were rel_i equals 1 if the item at position i is a relevant item (i.e. it belongs to the ground truth set) and 0 otherwise. And finally the *NDCG* is defined as:

$$NDCG = \frac{DCG}{IDCG} \tag{5.4}$$

If the intersection of G and R is empty, the DCG is 0 so the *NDCG* would be also 0.

- *Clicks.* In Spotify there is a feature called *Recommended Songs* that recommends 10 additional tracks given a set of tracks in a playlist. Every time the list is refreshed it produces 10 more tracks. The metric *clicks* represents the number of times the Recommended Songs list needs to be refreshed before a relevant track is shown to the user. It is calculated as follows:

$$Clicks = \left\lfloor \frac{\texttt{argmin}_\texttt{i}\{R_i : R_i \in G\} - 1}{10} \right\rfloor \tag{5.5}$$

  If there is no relevant track in R the maximum number of clicks is picked (which is 51).

The provided final value of each metric is the average of the individual metrics values obtained for each user playlist used in the evaluation.

5.2. Neighbour selection technique in CF

In memory based collaborative filtering the relevance of each item is calculated as a weighted sum of contributions from the set of users that are more similar to the target user. The set of similar users can be composed the top k more similar users (k nearest neighbours) or the users whose similarity metric is higher than a certain threshold. Parameters of both techniques, that are the threshold and the number of neighbours, were estimated using a grid search method. Both methods are similar but there are some particularities that may affect the final ranking:

1. k *Nearest Neighbours (CF-kNN).* Only the playlists with higher correlation values contribute to the final track ranking. The singularity of kNN method is that, after computing correlations between playlists, only the set of k playlists with higher correlation values are used to compute the recommendation. The final score of each track is the sum of the correlation

values of all playlists that contain that track divided by k. Tracks of the database are ranked in decreasing order of score values.The constant number of neighbours is a problem because the actual correlation is not taken into account. When there are more than k neighbours that are highly correlated to the user playlist the rest of them are discarded so there is potentially useful information that is neglected. In the opposite case when all playlists are low correlated to the user, the information that is used to rank tracks is not representative of the user preferences.

Neighbours	NDCG	Clicks	Precision	Aprox. Time/recom (s)
15	0.0047	46.94	0	10
20	0.0082	45.31	0.0006	10
25	0.0089	43.46	0.0032	10
30	0.0116	42.83	0.0034	10
35	0.0077	44.49	0.0007	10

Table 5.1. CF-kNN evaluation with neighbour numbers between 15 and 35

2. *Threshold filtering (CF-TH).* Playlists that have a minimum correlation to the user playlist are considered for the final track ranking. The number of playlists that reach that threshold is not constant, it changes depending on the user playlist. This method ensures that only playlists with a minimum similarity to the user are taken into account. The difficulty lies in the threshold selection. It should not be too restrictive so there is not enough information to build a good quality recommendation but it should also be low enough so to avoid playlists with little similarities to contribute to the recommendation. In figure 5.1 the cumulative distribution of playlists that on average are under a threshold is depicted. It can be seen that with thresholds higher or equal to 0.05 in average more than 88% of the playlists are discarded to compute the recommendation. On the other hand with thresholds higher or equal to 0.2 on average 99% of the playlists in the database

did not contribute to the recommendation. Considering that, the system was evaluated using thresholds from 0.05 to 0.3.

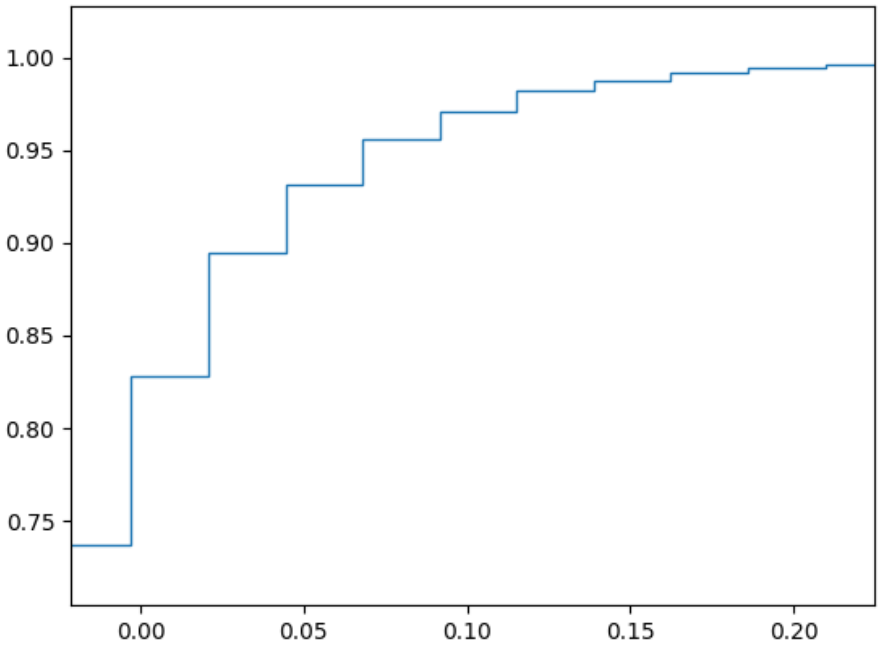


Fig. 5.1. Cumulative distribution of playlists below a certain threshold.

Threshold	NDCG	Clicks	Precision	Aprox. Time/recom (s)
0.05	0.0033	48.22	0	10
0.1	0.0052	47.24	0.0012	11
0.2	0.0042	47.18	0	9
0.3	0.0046	47.24	0	5

Table 5.2. CF-TH evaluation with thresholds between 0.05 and 0.3

Conclusions:

According to the obtained experimental results we can say that the threshold method did not provide good recommendations, since all of them show almost the worst metrics. The best results are the ones given by the threshold 0.1 but they are very close to the ones obtained with the other thresholds. Although the CF-kNN method did not obtained much better metrics statistics, it can be seen that metrics obtained by the CF-kNN method are better than the threshold ones. Metrics of the best case in CF-kNN evaluation are more than two times bigger that the threshold best case. In conclusion we can say that these two methods, in terms of number of relevant tracks recommended and ranking quality, do not provide quality recommendations the majority of the times, at least with the subset of the database we are using. However CF-kNN provided better recommendations specially for $k = 30$.

5.3. Content based techniques

Two main approaches were evaluated: distance similarity and GMMs. These decission was based on the comparison of several music content based recommendation approaches done by Bogdanov et. al. in [27] . They tested several algorithms, some of them using low level descriptors and others using high level descriptors. The approaches that had better results were the ones that used high level descriptors. Those high-level descriptors can be now obtained using Essentia extractors and were used in the evaluated content based approaches.

5.3.1. Distance based methods

Three variants using Pearson distance similarity were tested:

1. *Distance to the user mean feature vector (CB-DIST-MEAN).* The user is represented by a single point in the feature space. The point is calculated by averaging the feature of all the tracks in the user playlist. Then the tracks

in the recommendation list are the ones that are closer (more similar) to the user point in the feature space. Distance between the user (target) vector $\mathbf{x}_t$ and another track feature vector $\mathbf{x}_i$ is calculated using the Pearson correlation coefficient, and then the tracks with higher correlations are recommended. The main drawback of this method is that diversity of user preferences is simplified to only one average preference. In that case recommendations might suite worse the playlist content and the performance of the system could be worse.

2. *Top distance to individual tracks (CB-DIST-TOPI).* In this variant all distances between user tracks and tracks from the database are computed and then only the database tracks with lower distance to any of the user tracks are recommended. Distances are calculated using the Pearson correlation as mentioned before and then tracks are ranked according to their minimum distance to a track in the user list. This way the variability of the playlist is not simplified in a unique vector and the recommendation list can match different music styles present in the user playlist. It was used in [27] . This recommendation method depends on the distribution of the database along the different music styles. If an outlier of a user playlist is close to a group of songs in the feature space and the rest of the playlist is located in an area with very few close neighbours the majority of the recommendations will be more similar to the outlier. A real user might like those recommendations but our computational evaluation will worsen.

3. *Top average distance to individual tracks (CB-DIST-TOPA).* To reduce the impact of outliers while taking into account most of the diversity in the user list, this method rank tracks according to their average distance to all tracks in the user playlist. If there is a playlist which tracks are far and disseminated in the feature space all mean distances will be low but tracks that are closer to some of the user tracks will still have lower mean distance, so the diversity of styles present in the playlist will affect to the final recommendation list.

Name	NDCG	Clicks	Precision	Aprox. Time/recom (s)
CB-DIST-MEAN	0.0216	40	0	470
CB-DIST-TOPI	0.0419	27.4	0.0139	4036
CB-DIST-TOPA	0.0293	35.11	0	4041

Table 5.3. CB-DIST methods evaluation

5.3.2. GMM based methods

Three variants using GMMs were tested. In all of them a dimensionality reduction of the track representation is performed by PCA for several reasons:

- Descriptors taken from the same classifier might be highly correlated since they sum up to one. There are several multi class classifiers and using all classes may not give really unique and uncorrelated information.
- Playlists can have a minimum length of 20 tracks. The method used for building the GMMs from the library scikit-learn restricts the number of samples, so they must be greater or equal to the number of features. This is not a problem when the model is trained using 5 second segments of the tracks but otherwise that restriction affect directly the number of features that can be used.
- Reducing dimensions of track representation lightens the system so it takes less time to compute each recommendation.

The number of components were chosen analyzing the explained variance of the PCA. If the total variance is the sum of the variances of all principal components, the variance explained by a number of principal components is the sum of their variances over the total variance. In Figure 5.2 we can see that with 15 components more than 95% of the variance is explained. It is a very convenient number because even with the shortest playlists the restriction in the number of

features for training the GMMs is fulfilled. To evaluate the quality of the recommendations 20% of the tracks in the user playlist are taken, so the minimum number of tracks will be 16 instead of 20.

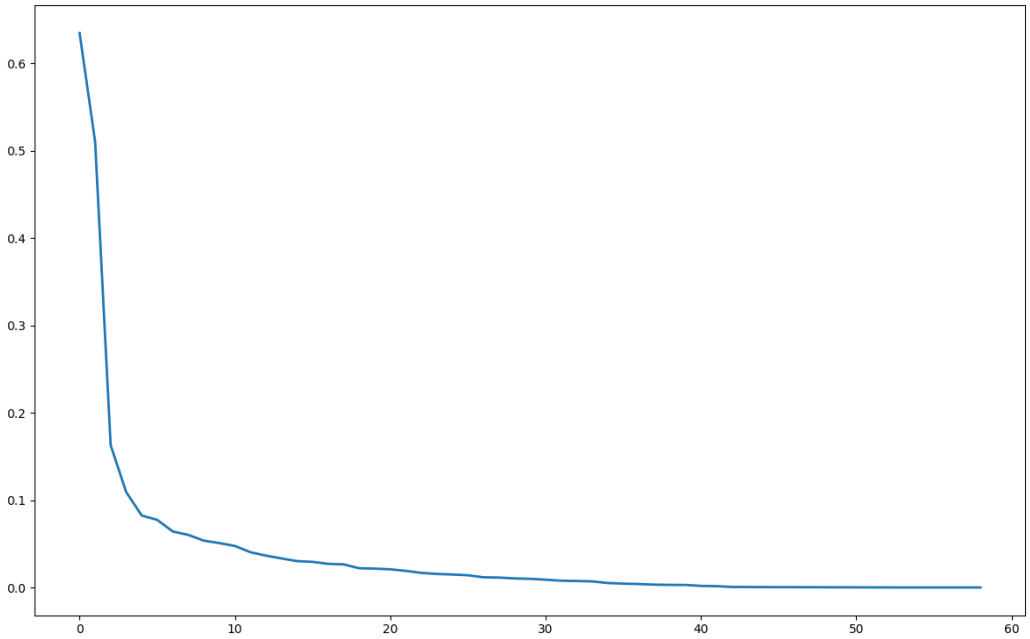


Fig. 5.2. Explained variance as a function of the number of components, computed over the entire database of tracks

1. *Gaussian Mixture Model (CB-GMM).* A GMM is trained with the tracks' feature vectors of the user playlist. As in [27] initial parameters are initialized using the k-means method. A grid search method was used to obtain the best number of mixture components. The evaluation of recommendations obtained for different numbers of mixture components is in Table 5.4

2. *GMM with sliced user playlist (CB-GMM-WIN).* Extracting the feature characterization of the signal in overlapping windows is a very used technique in speech recognition when working with GMMs. In this case with

N.Comp	NDCG	Clicks	Precision	Aprox. Time/recom (s)
1	0.0072	47.14	0.004	130
2	0.0013	48.3	0	129
3	0.0089	45.54	0.0032	129
4	0.0055	47.12	0.0028	128
7	0.0048	47.3	0	129
9	0.0090	42.7	0.0018	129
11	0.0043	47.34	0	129

Table 5.4. CB-GMM evaluation with number of mixture components between 1 and 11

30s tracks, we trained the user model with features extracted from 5s sections overlapped 1s. The purpose of this variant is to try to obtain more information about the user playlist so that the GMM can model it better. It was evaluated using 9 mixture components as it was the number of components for which the best metrics were obtained in the evaluation of CB-GMM.

N.Comp	NDCG	Clicks	Precision	Aprox. Time/recom (s)
9	0.0162	42.667	0	3454

Table 5.5. CB-GMM-WIN evaluation with number of mixture components equal to 9

3. *GMM with Universal Background Model (CB-GMM-UBM).* Using a UBM has shown to give good results in music similarity tasks [31]. The chosen number of components of the GMM is 9 because it was the number that gave better results in the valuation of CB-GMM. Init parameters of the UBM were initialized using k-means method.

N.Comp	NDCG	Clicks	Precision	Aprox. Time/recom (s)
9	0.0211	41	0.0125	260

Table 5.6. CB-GMM-UBM evaluation with number of mixture components equal to 9

Conclusions

Clearly distance-based methods CB-DIST-TOPI and CB-DIST-TOPA obtained better metrics. The reason behind this could be that if the characterization of the tracks is good then the distance between them is also a good measurement of their similarity and perhaps playlists tend to have several subgroups of tracks that are very similar between them but not necessary between other subgroups of the playlist. GMMs and CB-DIST-MEAN do not give that much importance to the individual variability of the tracks from the playlist. However CB-DIST-TOPI and CB-DIST-TOPA were the slowest methods because all distance combinations between playlist tracks and the whole database are computed. In a real database distances between tracks could have been pre-computed and stored so the actual time to compute recommendations can be highly reduced. If all distances need to be calculated, those two methods and CB-DIST-MEAN are not scalable since the number of distances that need to be computed is a multiple of the database size.

Comparing the GMMs variants (CB-GMM, CB-GMM-WIN and CB-GMM-UBM) we can say that while CB-GMM-WIN and CB-GMM-UBM increased the average time per recommendation, they did not obtained considerably better results than CB-GMM with the same number of mixture components. Specially CB-GMM-WIN needed much more time per recommendation because windowing all tracks of the user playlist and obtaining their corresponding feature representation was very slow.

5.4. Hybridization method

For improving the quality of the final recommendations two hybridization methods were tested. Because CF-kNN was the collaborative method that obtained better results it is the only collaborative method tested in combination with content-based systems. The hybridization methods that were evaluated are the following:

1. *Cascade.* The content-based recommender is given a list of track with length L by the collaborative system to compute the final recommendation, instead of using the whole database. This reduces the time of the content based recommendation, which is an important improvement since one of the main problems of that type of recommenders is that they can be considerably slow. Depending on the L collaborative filtering has more or has less importance for the final recommendation.

CB method	L	NDCG	Clicks	Precision	Aprox Time/recom(s)
CF-kNN + CB-GMM	500	0.014	40.14	0.001	33
CF-kNN + CB-GMM	1000	0.0222	36.1	0.0034	33
CF-kNN + CB-GMM	1000	0.021	38.17	0.0051	44
CF-kNN + CB-GMM	2000	0.018	38.08	0.0034	55
CF-kNN + CB-DIST-TOPI	100	0.0047	48.98	0	70
CF-kNN + CB-DIST-TOPI	300	0.0047	41.64	0	590
CF-kNN + CB-DIST-TOPA	100	0.0039	49	0	70
CF-kNN + CB-DIST-TOPA	300	0.0106	43.4	0.0028	590

Table 5.7. Cascade hybridization method evaluation were L is the length of the list of tracks that are taken into account in the conten-based method of the system

2. *Combination.* Recommendations are computed separately in a collaborative and a content-based system. Then the track ranking is built taking into account results given by each recommender. For the combination of kNN and GMM, the probability obtained in the content-based recommender is multiplied by a factor in the k tracks with higher correlation in the collaborative system. Combination factors were chosen inspecting the tracks probabilities given by each method. On average the difference between the highest probability and the lowest probability of the first 100 tracks in the collaborative recommendation list has 7 orders of magnitude. Considering that two combination factors were tested for the simplest GMM method. Only CB-GMM was tested .

Comb. Factor	NDCG	Clicks	Precision
100	0.0149	38.96	0.0011
1000	0.0188	39.41	0.0089

Table 5.8. HYB-COMB-GMM evaluation with combination factors 100 and 1000

Conclusions

CF-kNN + CB-GMM improved the results obtained by CB-GMM specialy for $L = 1000$ getting quite good metrics compared to other systems apart from CB-DIST-TOPI and CB-DIST-TOPA. In was also faster than CB-GMM so the improvement is substantial.

CF-kNN + CB-DIST-TOPA and CF-kNN + CB-DIST-TOPI were tested to try to reduce the average time per recommendation. However when the average time was more similar to the average time of CF-kNN + CB-GMM results worsened dramaticaly. For CF-kNN + CB-DIST-TOPI to have good results L should

be greater which implies that the average time would be also increased. So this hybridization did not improve CB-DIST-TOPA performance.

5.5. Human evaluation

Since recommender systems fundamental purpose is to provide recommendations to humans, an small survey was done to a group of 20 people. The main objective of this experiment was to compare human evaluations with objective metrics and see if there is a correlation between them so that they are a reliable measurement.

5.5.1. Survey details

The survey consisted in first listening several tracks from the user playlist and then evaluating the first 5 tracks of the recommendation list. The recommendation list is ordered according to the estimated relevance so the first 5 tracks are the ones the system considered they are more likely to match with the preferences of the user.

The hybrid system HYB-CA-GMM was the chosen system because it has been tested with a bigger number of playlists. After computing recommendations to 100 playlists four playlists were randomly selected for the human evaluation. Two playlist were selected from the group that was not recommended any relevant track and the other two were selected from the group of playliststhat was recommended any relevant item.

For practicality of the survey only a set of tracks from the users playlist were used in the evaluation instead of the whole playlist, because if would have taken too much time to be listened considering that the maximum length of a playlist in the database is 200.

People was asked to rate from 1 to 10 each recommended track according to two different criteria:

- Criterium 1: "Do you think the user will like this song?"

- Criterium 2: "Do you think the song is similar to the songs from the playlist?"

The first one is a more subjective question than the second one and tries to take into account the taste profile each person think the user has. The second one is only related to similarity, that can also be representative of the potential acceptance of the recommendation but not always, as shown in the comparison of content based approaches were the most successful system was the one that took more into account the variability of the user playlist.

5.5.2. Final results of the survey

Test number	NDCG	Clicks	Precision	Mean rating
1	0	51	0	6.2
2	0.07	4	0	7.8
3	0	51	0	4.3
4	0.3	0	0.16	8.9

Table 5.9. Human mean rating on each test compared to objective metrics.

Metrics are very different to the true average value of the ratings obtained in the human evaluation, however we can say that both evaluations are correlated. The average ratings were higher in the playlists that obtained better metrics and lower on the ones that obtained the worst metrics.

6. Conclusions

6.1. Conclusions

Several types of recommender systems with a range of complexity levels were compared: memory-based collaborative systems, two types of content-based systems and hybrid systems. Also criteria and methods that were used to choose the architecture of the systems and for parameter optimization were provided.

An objective evaluation process was done to test all the considered methods. The chosen metrics took into consideration the number of relevant recommendations as well as the ranking quality of the recommendation list. This process concluded obtaining two different options of music recommender system architectures. Those final systems provided satisfactory recommendations according to the objective metrics.

If we take into account the time needed for the recommendation, the most functional option was an hybrid recommender that is composed of two methods in cascade. A kNN filtering method in the first stage and a GMM to compute the final recommendation.

The system that obtained better results was a content-based system that recommends the tracks that are closer to any of the user playlist tracks in the feature space generated by the high-level features provided by Essentia. The drawback of this system is the time needed for the computation.

After providing an overall view of recommender systems, analysing the special case of music recommendation and evaluating different approaches of music recommender systems, two architectures were proposed as possible solutions to the problem of music recommendation for user playlists.

6.2. Future work

There are some possibilities that were not evaluated and could be interesting to compare to the options already tested or that could have obtained better results:

- Model-based collaborative filtering was not tested. More complex collaborative filtering methods might have better results than the one tested.
- Only one recommender was evaluated by humans. Since objective metrics and human ratings were so different it could be interesting to compare the two evaluation techniques with more recommendation algorithms.
- A real recommendation evaluation experiment could be done to obtain more useful evaluations. Rating other users' recommendations is not necessary representative of what a real user would have evaluate.
- Proposed systems were not tested with a different database. A database with playcounts or other types of user behaviour information could affect the performance of the recommender systems. Evaluating if the systems are directly exportable to other databases might be of some interest.
- Only Essentia high-level descriptors were used. To evaluate if semantic descriptors are more useful for the recommendation task lower lever features could be tested, for example MFCCs.

Bibliography

[1] F. Isinkaye, Y. Folajimi, and B. Ojokoh, "Recommendation systems: Principles, methods and evaluation," *Egyptian Informatics Journal*, vol. 16, no. 3, pp. 261–273, 2015.

[2] P. Lops, M. De Gemmis, and G. Semeraro, "Content-based recommender systems: State of the art and trends," in *Recommender systems handbook*, pp. 73–105, Springer, 2011.

[3] M. Schedl, E. Gómez, and J. Urbano, "Music information retrieval: Recent developments and applications," vol. 8, pp. 127–261, 01 2014.

[4] M. Schedl, H. Zamani, C.-W. Chen, Y. Deldjoo, and M. Elahi, "Current challenges and visions in music recommender systems research," *International Journal of Multimedia Information Retrieval*, vol. 7, pp. 95–116, Jun 2018.

[5] K. Verstrepen, K. Bhaduriy, B. Cule, and B. Goethals, "Collaborative filtering for binary, positiveonly data," *SIGKDD Explor. Newsl.*, vol. 19, pp. 1–21, Sept. 2017.

[6] J. Lee, M. Sun, and G. Lebanon, "A comparative study of collaborative filtering algorithms," *CoRR*, vol. abs/1205.3193, 2012.

[7] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, (New York, NY, USA), pp. 285–295, ACM, 2001.

[8] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "Grouplens: An open architecture for collaborative filtering of netnews," pp. 175–186, 1994.

[9] C. Desrosiers and G. Karypis, "Content-based recommender systems: State of the art and trends," in *Recommender systems handbook*, pp. 107–144, Springer, 2011.

[10] X. Amatriain, A. Jaimes*, N. Oliver, and J. M. Pujol, *Data Mining Methods for Recommender Systems*, pp. 39–71. Boston, MA: Springer US, 2011.

[11] A. Gershman, A. Meisels, K.-H. Lüke, L. Rokach, A. Schclar, and A. Sturm, "A decision tree based recommender system," in *IICS*, pp. 170–179, 2010.

[12] R. Burke, "Hybrid recommender systems: Survey and experiments," *User Modeling and User-Adapted Interaction*, vol. 12, pp. 133–370, 11 2002.

[13] D. Billsus and M. J. Pazzani, "A hybrid user model for news story classification," in *UM99 User Modeling*, pp. 99–108, Springer, 1999.

[14] I. T. Jolliffe, "Graphical representation of data using principal components," *Principal component analysis*, pp. 78–110, 2002.

[15] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[16] J. S. Downie, "Music information retrieval," *Annual review of information science and technology*, vol. 37, no. 1, pp. 295–340, 2003.

[17] B. Manjunath, P. Salembier, and T. Sikora, "Introduction to mpeg-7: Multimedia content description interface," 01 2002.

[18] M. Worring, A. W. Smeulders, A. Gupta, S. Santini, and R. Jain, "Content-based image retrieval at the end of the early years," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 22, pp. 1349–1380, 12 2000.

[19] F. Gouyon, *A computational approach to rhythm description — Audio features for the computation of rhythm periodicity functions and their use in tempo induction and music content processing*. PhD thesis, Universitat Pompeu Fabra, 2005.

[20] M. Leman, L. Clarisse, B. De Baets, H. De Meyer, M. Lesaffre, G. Martens, J.-P. Martens, and D. Van Steelant, "Tendencies, perspectives, and opportunities of musical audio-mining," in *Forum Acusticum Sevilla*, vol. 33, pp. 3–4, 2002.

[21] G. Peeters, "A large set of audio features for sound description (similarity and classification) in the cuidado project," 01 2004.

[22] F. Ernawan, N. Abu, and N. Suryana, "Spectrum analysis of speech recognition via discrete tchebichef transform," vol. 8285, pp. 82856L–82856L, 10 2011.

[23] B. Logan, "Mel frequency cepstral coefficients for music modeling," 11 2000.

[24] S. S. Stevens, J. Volkmann, and E. B. Newman, "A scale for the measurement of the psychological magnitude pitch," vol. 8, pp. 185–190, 01 1937.

[25] R. Loughran, J. Walker, M. O'Neill, and M. O'Farrell, "The use of mel-frequency cepstral coefficients in musical instrument identification," in *International Computer Music Conference*, International Computer Music Association, 2008.

[26] J. H. Jensen, M. G. Christensen, M. N. Murthi, and S. H. Jensen, "Evaluation of mfcc estimation techniques for music similarity," in *2006 14th European Signal Processing Conference*, pp. 1–5, Sept 2006.

[27] D. Bogdanov, M. Haro, F. Fuhrmann, E. Gómez, and P. Herrera, "Content-based music recommendation based on user preference examples," in *The 4th ACM Conference on Recommender Systems. Workshop on Music Recommendation and Discovery (Womrad 2010)*, (Barcelona, Spain), 26/09/2010 2010.

[28] D. Bogdanov *et al.*, "From music similarity to music recommendation: Computational approaches based in audio features and metadata," *PhD. Universitat Pompeu Fabra, Barcelona, Spain*, 2013.

[29] U. Shardanand and P. Maes, "Social information filtering: algorithms for automating "word of mouth"," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 210–217, ACM Press/Addison-Wesley Publishing Co., 1995.

[30] F. Pachet and J.-J. Aucouturier, "Improving timbre similarity: How high is the sky," *Journal of negative results in speech and audio sciences*, vol. 1, no. 1, pp. 1–13, 2004.

[31] C. Charbuillet, D. Tardieu, G. Peeters, *et al.*, "Gmm supervector for content based music similarity," in *International Conference on Digital Audio Effects, Paris, France*, pp. 425–428, 2011.

[32] B. Logan and A. Salomon, "A music similarity function based on signal analysis," in *IEEE International Conference on Multimedia and Expo, 2001. ICME 2001.(ICME)*, vol. 00, p. 190, 08 2001.

[33] D. A. Reynolds, R. C. Rose, *et al.*, "Robust text-independent speaker identification using gaussian mixture speaker models," *IEEE transactions on speech and audio processing*, vol. 3, no. 1, pp. 72–83, 1995.

[34] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, "The million song dataset," in *ISMIR 2011: Proceedings of the 12th International Society for Music Information Retrieval Conference, October 24-28, 2011, Miami, Florida*, pp. 591–596, University of Miami, 2011.

[35] D. Bogdanov, N. Wack, E. Gómez, S. Gulati, P. Herrera, O. Mayor, G. Roma, J. Salamon, J. R. Zapata, and X. Serra, "Essentia: an audio analysis library for music information retrieval," in *International Society for Music Information Retrieval Conference (ISMIR'13)*, (Curitiba, Brazil), pp. 493–498, 04/11/2013 2013.

[36] S. Streich and P. Herrera, "Detrended fluctuation analysis of music signals danceability estimation and further semantic characterization," in *AES 118th Convention*, 2005.

[37] M. A. Hall, “Correlation-based feature selection of discrete and numeric class machine learning,” 2000.

[38] T. Westerveld, A. de Vries, and F. de Jong, *Generative Probabilistic Models*, pp. 177–198. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.

[39] X. Hu and J. Downie, “Exploring mood metadata: Relationships with genre, artist and usage metadata.,” in *Proceedings of the 8th International Conference on Music Information Retrieval, ISMIR 2007*, pp. 67–72, 01 2007.

Annex

Low-level descriptors

List of all low-level descriptors extracted by the Essentia music extractor [35]

Low descriptors

Complete list of low-level descriptors belonging to the subclass 'Low' in alphabetical order:

- Average Loudness
- Barkbands crest
- Barkbands flatness
- Barkbands kurtosis
- Barkbands skewness
- Barkbands spread
- Barkbands
- Dissonance
- Dynamic complexity
- ERB bands crest
- ERB bands flatness
- ERB bands kurtosis
- ERB bands skewness
- ERB bands spread
- ERB bands
- GFCC
- HFC
- Melbands crest
- Melbands flatness
- Melbands kurtosis
- Melbands skewness
- Melbands spread
- Melbands
- MFCC
- Pitch salience

- Silence rate 20dB
- Silence rate 30dB
- Silence rate 60dB
- Spectral centroid
- Spectral complexity
- Spectral contrast coeffs
- Spectral contrast valleys
- Spectral decrease
- Spectral energy
- Spectral energyband high
- Spectral energyband low
- Spectral energyband middle-high
- Spectral energyband middle-low
- Spectral entropy
- Spectral flux
- Spectral kurtosis
- Spectral rms
- Spectral rolloff
- Spectral skewness
- Spectral spread
- Spectral strong peak
- Zero-crossing rate

Rhythm descriptors

Complete list of low-level descriptors belonging to the subclass 'Rhythm' in alphabetical order:

- Beats count
- Beats loudness
- Beats loudness band ratio
- Beats position
- bpm
- bpm histogram first peak bpm

- bpm histogram first peak spread
- bpm histogram first peak weight
- bpm histogram second peak bpm
- bpm histogram second peak spread
- bpm hist. second peak weight
- Danceability
- Onset rate

Rhythm descriptors

Complete list of low-level descriptors belonging to the subclass 'Tonal' in alphabetical order:

- chords histogram
- chords key
- chords number rate
- chords scale
- chords strength
- hpcp entropy
- hpcp
- key (key)
- key (scale)
- key strength

- tuning diatonic strength
- tuning equal tempered deviation
- tuning frequency
- Tuning nontempered energy ratio

High-level descriptors

Additional information about semantic descriptors extracted by Essentia high level extractor

5 mood cluster classifier

A set of five mood clusters was proposed in [39] with the intention of reduce the mood space into a smaller and more manageble set. Each cluster is defined by several words, as shown in the following table 6.1:

Cluster 1	**Cluster 2**	**Cluster 3**	**Cluster 4**	**Cluster 5**
Rowdy	Amiable/Good natured	Literate	Witty	Volatile
Rousing	Sweet	Wistful	Humorous	Fiery
Confident	Fun	Bittersweet	Whimsical	Visceral
Boisterous	Rollicking	Autumnal	Wry	Aggressive
Passionate	Cheerful	Poignant	Quirky	Intense

Table 6.1. Words that define the mood spaces of the 5 mood clusters.

Accuracies of the classifiers

Classifier	Classes	Accuracy
G1	Alternative, blues, electronic, folk/country, funk/-soul/rnb, jazz, pop, rap/hiphop, rock	60.25%
G2	Classical, dance, hip-hop, jazz, pop, rhythm 'n blues, rock, speech	87.55%
G3	Blues, classical, country, disco, hip-hop, jazz, metal, pop, reggae, rock	75.52%
GEL	Ambient, drum 'n bass, house, techno, trance	91.69%
CUL	Western, non-western	93.47%
MHA	Happy, non-happy	84.90%
MSA	Sad, non-sad	87.82%
MAG	Aggressive, non-aggressive	97.50%
MRE	Relaxed, non-relaxed	93.20%
MAC	Acoustic, non-acoustic	92.98%
MEL	Electronic, non-electronic	86.38%
MCL	5 mood clusters	57.08%
RPS	Slow, medium, fast	77.64 %
RBL	Chachacha, jive, quickstep, rumba, samba, tango, Viennese waltz, waltz	73.20%
ODA	Danceable, non-danceable	92.41%
OPA	Party, non-party	88.38%
OVI	Voice, instrumental	93.8%
OTN	Tonal, atonal	97.67%
OTB	Bright, dark	94.31%
OGD	Male, female	87.21%

Table 6.2. High-level descriptors and accuracies of the classifiers. Source: `http://essentia.upf.edu/documentation/svm_models/accuracies_v2.1_beta1.html`.