POLITECNICO DI MILANO
Facoltà di Ingegneria dell'Informazione

POLO REGIONALE DI COMO

Master of Science in
Computer Engineering

MUSIC RECOMMENDER SYSTEM

Supervisor: Prof. Sara Comai

Master Graduation Thesis by: Carlos Álvarez Angulo
Student Id. number 737815

Academic Year 2010/11

Abstract

The growing use of Internet as an information source, has led to the proliferation of technologies to deploy rich web based applications. Among these applications are present the music service providers. These systems allow users to listen to music without downloading it to the computer. Some use recommendations techniques to improve the user experience.

The objective of this project is to develop a music recommendation system. The system will determine the musical preferences of the users based on the analysis of their interaction during use. This way the system is able to estimate what artist or group would match user preferences to the user at a given time. It has been taken into account the fact that we do not always want to hear the same artists or genres, we do have favorite bands, but sometimes we appreciates a surprise, a new discovery.

The system uses music information collected from online music services that make available their music catalogs for developers' community to be used inside new applications. It has been created a web system that connects to the music service providers to obtain these musical catalogs. This system implements the necessary communication features to use this information in the client web browser.

This system helps users discover new artists, albums or songs making the musical catalog available for listening. The dynamic characteristics of the interface allows the user to browse music collections while listening to a song or playing a video. The user will receive information related to her interaction patterns in form of recommendations of items. These items will probably match user preferences and they are shown as the user interacts with the system and only when it has enough information about user preferences.

1. Introduction

The Internet evolution continuously generates several changes in social habits related to communication and lifestyle. The bandwidth growing originated the birth and late spreading of complex file-sharing systems. This systems known as peer to peer software let users share files they had stored locally in their personal computers with other users connected to the same system. Music sharing started thanks to software like Napster (www.napster.com) or late Audiogalaxy (www.audiogalaxy.com). These peer to peer systems revolutionized the music industry and so the habits of people related to musical collect and playback. Now it was easier to search music, easier to store music, and much cheaper to get it. This new situation led to massive music storage for sharing purposes and affected the way music was reproduced, changing from complete and straight album reproduction to the creation of complex playlists composed of many artist and musical genres.

The continue increasing in connection speed and trends in web development technologies, given rise to large web systems nowadays visited daily. Among these advanced systems, there are systems that allow users to listen music online without the need of downloading it to their personal computer. This issue solves a big problem originated by peer to peer software. This is the music copyright problem confronted with the music purchaser rights. The time to define the line delimiting the freedom a user has to share something he allegedly bought legally had come. Big music distribution companies started legal battles against the most important peer to peer software owners, the success of these legal processes depended on the copyright laws in each peer to peer hosting country. Despite some peer to peer software systems stand till nowadays, this web music services came up as new music sharing formulas.

The music listening services own big music catalogs, in order to provide a wide public use. These same services manage the copyright problems for each country. They adapt the musical catalog according the copy and reproduction rights of the musical label associated to each album. Most of these music services are paid, some provide free access to the musical catalog, but no reproduction rights. There is a wide variety of these systems and new alternatives are constantly emerging increasingly improved. Some are simple players providing playlist functionality (prostopleer.com), others accompany the player with a recommendation system of similar artists (www.spotify.com), also there are complex collaborative systems in which hundreds of people leave comments on songs (www.pandora.com, www.lastfm.com) and have the chance of interact with each other as in the newly emerging social networks.

Music Recommender systems can be seen as a surrogate of real-world radio stations and music magazines. These real-world organizations main purpose is to promote certain artists, sometimes because the radio directors or magazine editors find noteworthy the quality of their musical works, sometimes just because of economic interest. Some people listen to these stations and read magazines in order to make decisions about what music to acquire:

either by traditional means or through some share-alike network. Music recommenders have the chance of making accessible to users not only the market-defined “good music”, but also new emerging groups, minor rare music and independent label’s productions.

Approaches to recommendation

Recommendation is an important field strongly related to web business which has been intensely researched in the past years, since electronic commerce web sites started their activity. Among the reviewed approaches, many solutions were found for data analysis, data colleting, or data-objects representation. According these, models like content-based, collaborative or context-based give differenced solutions to select key information to face recommendations. In parallel to the recommender model’s development also evolves the mathematical world related to the most pure heuristic bases of recommender systems.

The most common mathematical [1] models used in current recommender systems have been reviewed helping the author of this project to build up a solid idea about what recommendation is and how can it be achieved. These mathematical approaches to recommenders:

- *Logic recommender systems* [17] try to find an exact match among the recommendation options compared the user profile. The data representation is build using the attributes that define objects. Attribute types as used as they are with no further abstraction.
- *Vector space-based systems* [28] due to its numeric data modeling, estimate which objects best suit the user profile statements. Data object *surrogates* represent attributes in vector form being each cell a concrete attribute.
- *Probabilistic systems* [17] estimate a concrete object’s importance using a probability function. This function estimates the probability the object has in order to meet the preferences stated in the user profile.

The application of these methods depends on the features of the recommendation problem itself. An important step of the design phase is how to adapt the problem abstraction to a suitable mathematical model,

This project has been designed in order to meet some logic-based recommender features. In one hand, due to the need of overcoming the numerical representation of data, needed to use other mathematical approaches, in other hand because is allows the closest data representation domain to this problem.

Project definition

The objective of this project is to develop a music recommendation system. The system will determine the musical preferences of users based on the analysis of their interaction during use. This way it's possible to estimate what artist or group would be pleasing to the user at a given time. It has been taken into account that we do not always want to hear the same artists or genres, we have favorite bands, but sometimes a listener needs to be surprised, enjoy a new discovery. A music recommender needs a music catalog to be recommended, this project offers a review of several music services and their features, making use of musical information available online provided by some (friendly) music services. These services allow developers to access their musical collection to contribute with the proliferation of new applications. It has been created a web component that connects to music service providers to obtain this data. The web system implements the essential communication skills to use this information within the client web browser.

This system helps users discover new artists, albums or songs making this music information accessible. For doing this, a complete analysis of the state-of-the-art for music recommendation has been performed, giving clear highlights about the recommendation techniques used in many systems which implement recommendation thought diverse mathematical models. The dynamic characteristics of the interface allows the user to browse music collections while listening to a song, album, or playing a video. The users will receive information related to their interaction patterns (profiles) as personalized recommendations of items which probably they would like, while they use the application.

Goals and scope outline

The goals of this project are outlined below:

- Create a music recommendation system able to infer the user's musical preferences in a given time. The scope is not to know the user; instead it's about estimating what he could like right now.
- Explore the music services available nowadays looking for a complete and freely accessible music catalog and free streaming services.
- Develop a working system capable of making the most of free online services to provide the user with a completely free system which brings the opportunity of discovering new music.

Topics which won't be covered:

- This application is not for commercial use. So many issues related to it, like international translation or fine browser compatibility are out from this work outlines.
- The intention of this work in not to create a big music recommender or a big music streaming web site. The application itself stands as a data collector and as a benchmark for checking if the data obtained from web services is valid to be used for recommendation purposes.

Thesis structure

This project is composed by two parts of differenced nature. First it has been performed an introductory review about recommendation, explaining why the recommender systems are so important for electronic commerce, what kind of systems do perform recommendation focusing arguments in music recommendation. Along the section 2 of this document, it's presented a general recommender system's overview, including a possible taxonomy [6], a technical overview [17] and a classification [1].

Section 3 includes a complete music recommender's overview, including previously proposed approaches, and reviews of commercial and non-commercial systems which are currently online. Following this introduction to music recommenders, the problem this project is facing is defined (Section 4) and so the selected methodology to achieve it (Section 5). The development has been sliced in three iterations, which faithfully represent the real project evolution. Sections 6.2, 6.3 and 6.4 deeply explain the actions and decisions taken in each one.

After presenting this project, the conclusions' section (7) includes some observations inferred from the whole project execution. The contributions achieved in this project, are *just my two cents* for the huge knowledge about recommendation out there.

2. The recommender systems

2.1. Introduction

The roots of recommender systems were settled due to special needs of works in diverse fields: cognitive science [19], information retrieval [20] or economics [21]. Recommender systems emerged as an independent research area in the middle 90s and their important role to enhance data accessibility attracted the attention of both, academic and industrial worlds.

Recommender systems are a useful way to expand search algorithms since they help users discover items they might not have found by themselves. A recommendation is basically to present the user with some items which would match his preferences. There exist different approaches [1] to collect information about the users, by monitoring their interaction, by asking them to perform some actions or to fill some forms with personal information. The user's interaction with the system provides two types of information:

Implicit information: Collected from the user interaction itself. For example, by keeping the items the user has interacted with, and item related information like viewing times, item's reproductions or user related information as group membership.

Explicit information: The users provide this information every time they give opinion about items, rating or liking some item. Generally all the information elaborated by the user consciously.

The recommender system collects both kinds of information to generate the user profile. This profile stores information not only about the user likes, also information about the user itself, current placing, current personal needs, sex, age, professional position, and so. The way it's used by the recommendation system varies a lot among the different systems. The information stored within is also a determinant factor in the recommender algorithm design.

2.2. Taxonomy for recommender systems

A possible taxonomy of the recommender systems it has been proposed in [1]. The categories in which is divided describe diverse models of abstraction for user profile, how it is generated, and how is it late maintained and how does it evolve as the system runs.

User profile representation: An accurate profile is an important task since the recommendation success depends on how the system represents the user's interests. Next are listed some models applied in current recommender systems:

- *History-based*

Some systems keep a list of purchases, the navigation history or the content of e-mail boxes as a user profile. Additionally, it is also common to keep the relevant feedback of the user associated with each item in the history. Amazon[1] web site is a clear example.

- *Vector-space*

In the vector space model, items are represented with a vector of features, usually words or concepts which are represented numerically as frequencies, relevance porcentaje or probability.

- *Demographic*

Demographic filtering systems create a user profile through stereotypes. Therefore, the user profile representation is a list of demographic features which represent the kind of user.

- *User-item ratings matrix*

Some collaborative filtering systems maintain a user-item ratings matrix as a part of the user profile. The user-item ratings matrix contains historical user ratings on items. Most of these systems do not use a profile learning technique. Systems like Jamendo[2] include this technique to represent user profile.

- *Classifier-based models*

Systems using a classifier as a user profile learning technique, elaborate a methodology to monitor continuously input data in order to classify the information. This is the case of neural networks, decision trees and Bayesian networks.

- *Weighted n-grams*

Items are represented as a net of words with weights scoring each linking between nodes. For example in [22]), the system is based on the assumption that words tend to occur one after another a significantly high number of times, extracts fixed length consecutive series of *n* characters and organizes them with weighted links representing the co-occurrence of different words. Therefore, the structure achieves a context representation of the words.

[1] www.amazon.com
[2] www.jamendo.com

Initial profile generation:

- *Empty*: the profile is built as the users interact with the system.
- *Manual*: the users are asked to register their interest beforehand.
- *Stereotyping*: Collecting user-related information like city, country, lifestyle, age or sex.
- *Training set*: providing the users with some items among which they should select one.

Profile learning technique: The way the profile changes during time.

- *Not needed:* Some systems do not need profile learning technique. Some because they load the user related information from a database or it's dynamically generated.

- *Clustering:* Is the process of grouping information objects regarding some common features inherited to its information context. User profiles are often clustered in order to groups according to some rule. To assess which users share common interests. Recommenders like Last.fm[3] or iRate[4] perform this technique [12].

- *Classifiers:* Classifiers are general computational models for assigning a category to an input. To build a recommender system using a classifier means using information about the item and the user profile as input, and having the output category represent how strongly to recommend an item to the user. Classifiers may be implemented using many different machine learning strategies including neural networks, decision trees, association rules and Bayesian networks [1].

- *Information retrieval techniques:* When the information source has no clear structure, pre-processing steps are needed to extract relevant information which allows estimation of any information container's importance. This process comprises two main steps: feature selection and information indexing.

Relevance feedback: The two most common [1] ways to obtain relevance feedback is to use information given explicitly or to get information observed implicitly from the user's interaction. Moreover, some systems propose implicit-explicit hybrid approaches.

- *No feedback:* Some systems do not update the user profile automatically and, therefore, they do not need relevance feedback. For example, all the systems which update the user profile manually.
- *Explicit feedback:* In several systems, users are required to explicitly evaluate items. These evaluations indicate how relevant or interesting an item is to the user, or how relevant or interesting the user thinks an item is to other users. Some systems invite

[3] www.last.fm
[4] irate.sourceforge.net

users to submit information as track playlists. iRate uses this approach to provide its recommender with finer information about user's preferences.

- *Implicit feedback:* Implicit feedback means that the system automatically infers the user's preferences passively by monitoring the user's actions. Most implicit methods obtain relevance feedback by analyzing the links followed by the user, by storing a historic of purchases or by parsing the navigation history.

2.3. Design paradigms for recommender systems

One of the most practical issues for designing optimal-response recommendation systems is to represent and ontology the actors involved in the recommendation process in a suitable way for the selected model.

A general model for recommender system design optimally explained in [11] provides an interesting overview of this kind of systems. Next diagram [fig 1] makes a clear idea of this general model.

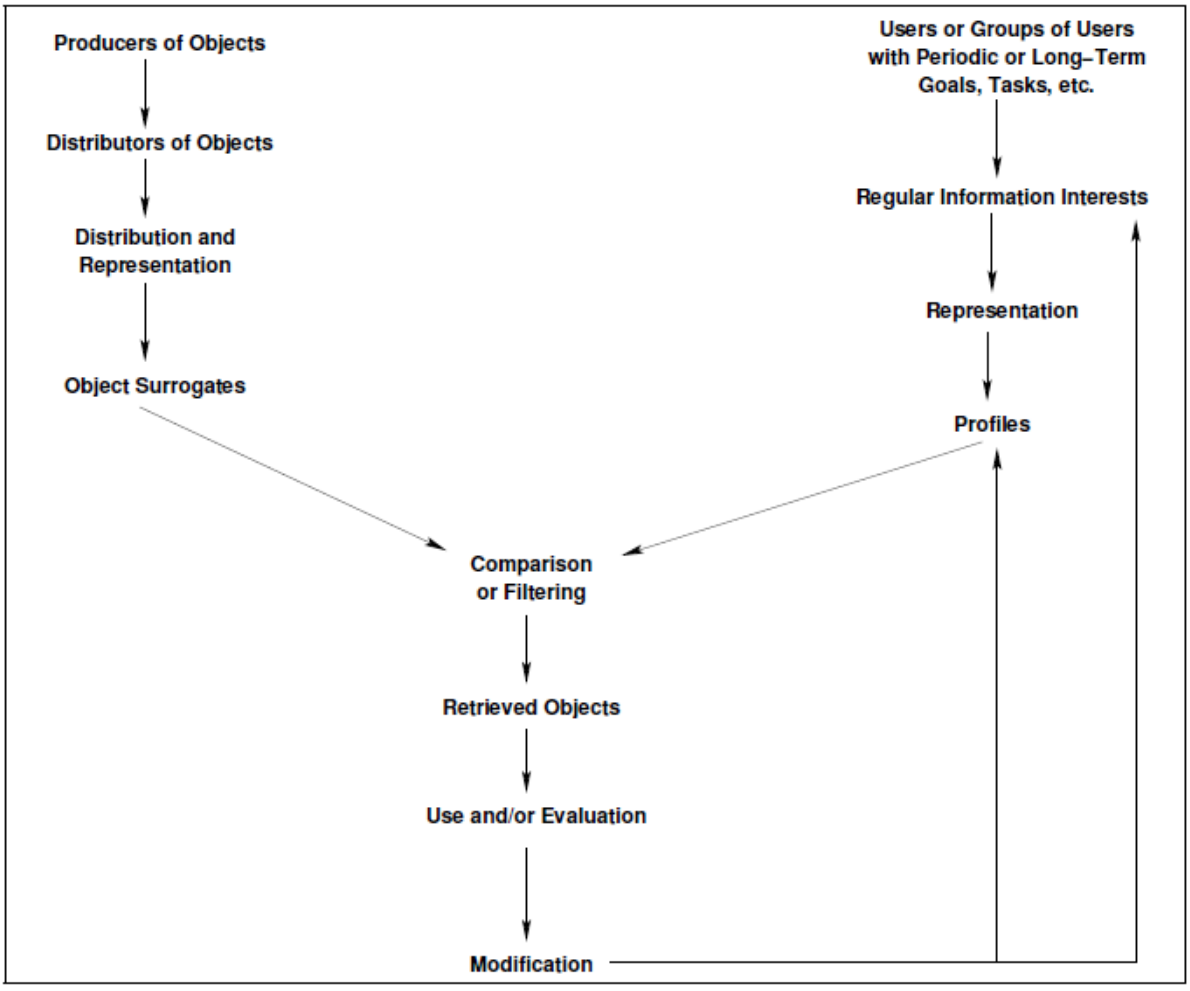

fig 1 General model for recommender systems

Design paradigms will provide solutions for assessing, the domain representation for actor *surrogates* and for algorithms the system may use over this *surrogates* to infer their utility. These design paradigms have been optimally reviewed in [17]. It is convenient to concretely determine the domain for the recommender system. This domain, in order to be a useful starting point for system design, should model or represent the following entities and processes:

Objects of interest

These are the objects subject to evaluation which stand as main information units the system will use to make recommendations. For a music recommender, these objects might be artist, albums and tracks.

Users

Each user has a user profile modeled following some design pattern depending on the design paradigm chosen for the system. This profile is the item against the objects of interest will be compared. The profile is the representation of the user preferences, related somehow to the objects the user could consider interesting. So the user profile will be a representation built with a set of item *surrogates* or with some generalization of those.

User environment

More transient information about the user, like local time and day, the task in which the user is involved currently, user mood, etc. It shortly amplifies the user profile expressiveness.

It should be noted that the performance of the frameworks presented next, will suffer severely due to limitations in the scope of available features. For instance, when item *surrogates* do not feature attributes that represent some the actual features people perceives on those objects. Obviously, this is not a problem of the frameworks themselves, but about the manifestation of the knowledge-representation designers.

2.3.1. Logic Recommender systems

This model is based on the idea of *exact match*: the system rejects or accepts objects depending on whether they satisfy the constraint statements present in the user profile. As described in [17], objects that match the constraints in the profile, since share the same characteristics, are considered to have the same utility value.

Domain representation

The object *surrogates* are represented as a group of attributes associated to some object identifier or to some description text. Each of these attributes has a well-defined type, which fully conveys the semantics of the attribute value. Regarding user profiles, those

collections of statements that define which attributes values are considered useful. These statements effectively restrict the range of values an object's attribute may take so that it is considered useful.

Comparison process

The main operation for comparison is checks whether the set of attributes associated to a given object satisfy the constraints encoded in the user profile. It is searched within an object repository which matches perfectly those constraints. The number of attributes to consider is fixed. This process turns each list of attributes attached to indexed objects into a list of boolean assessments that represent whether a constraint present in the profile is satisfied or not.
Usually, to avoid too strict specification, the list of constraints is parsed in disjunctive normal form: as soon as a constraint is satisfied, the object is accepted.

Drawbacks and limitations

The profile expressiveness is acquired explicitly by making the system ask users about their long-term information needs, this is forcing these users to express themselves. Then the reliability of the information conveyed by a profile might be suspicious, allowing the possibility of mistaken answers encoded as correct.

2.3.2. Vector Space-Based Recommender systems

These systems profit multidimensional properties of vector-spaces to represent item's *surrogates*. The *surrogates* must be numerically performable in order to be placed inside a multi-dimensional environment. In [28] it is explained an approach to recommendation using this model.

Domain representation

As the name of this model suggests, domain entities *surrogates* are vectors, where each dimension represents a certain object attribute. Attribute type should be real numbers.
User profiles represent features which satisfied certain used needs in the past. Objects whose *surrogates* are found to be similar to those appearing on the user profile, are assumed to have the a utility value proportional to the degree of similarity they share.

Comparison process

While the boolean model tried to compute exact match, vector-space model aims at computing the degree of similarity between object *surrogates* and the set of prototypes specified within the user profile. Item *surrogates* can be compared using vector-space related methods. In order to compare two real-valued vectors, the Euclidean distance or Cosine of the angle, method provides reliable vector matching [24].

Drawbacks and limitations

Vector space models do not allow to directly employ categorical attributes in the object *surrogates*. In order to do so, these categorical variables must be mapped to a subset of the real line. And this mapping function must preserve the ordering relationship, if any, in the value set. Sometimes, these variable semantics are not amenable to be expressed as an ordered set of numbers, leading to leave out these, since the framework does not offer support for them.

2.3.3. Probabilistic Recommender systems

Domain representation

User preferences are represented as a probability distribution. Object's *surrogate* attributes become the variables for this probability distribution. Therefore, the problem is to estimate the parameters for the probability distribution that maximize the likelihood of observed user behavior. As [17] shows this gives an interesting twist to the problem of modeling long-term user information needs, since user profile can be seen as a stochastic process that produces statements assessing which objects are interesting.

The probabilistic framework, in principle, allows integration of both numeric and categorical observations.

Comparison process

The uncertainty hypothesis for recommender systems relies on whether an object belongs or not to the set of objects that satisfy user's information needs. This hypothesis is calculated for each object element's surrogate .

The probability is calculated as follows:

This inference method, in contrast with the previously discussed, requires to estimate the parameters that define , also called likelihood of , as well as , called hypothesis prior. is not relevant for the problem of interest, because it is based on the interaction user needs which are not required to solve recommendation . The most usual way to estimate is applying the Maximum Likelihood estimation[5] procedure, deeply explained in [27]. This estimation requires to have available a set of *surrogates* for objects that are known to be in the relevant set.

[5] mercury.bio.uaf.edu/courses/wlf625/readings/MLEstimation.pdf

Drawbacks and limitations

In practice numeric attributes can be difficult to handle since they imply integrating the estimated probability density function, as stated in [24]. Numeric attributes can be clustered, or better said, encoded as a set of discrete symbols. These resulting synthetic categorical attributes are not the exact equivalent of their numeric meaning, because any mapping function from the infinite set of numbers to a finite, not very large set of integer numbers implies a loss of information. The assignment of points to target symbols and the measurement of the associated distortion phenomena is optimally obtained if:

- Maximizing the homogeneity of assignments
- Maximizing the minimum separation between cluster assignees

This fact is considered a NP-hard problem[6] since no exact, polynomial solutions are known. Therefore lots of numerical approximations to this problem have been analyzed as optimally reviewed in [18].

[6] www.math.ohiou.edu/~just/bioinfo05/supplements/Lect_NP.ppt

2.4. Classification for recommender systems

Not general agreement about classification for recommenders it has been found while reviewing previous works. The recommendation systems were optimally classified in [6] by sorting the system's recommendation approach in a quite generalized overview. The way in which the recommendation is faced in terms of scope:

The **heuristic based** techniques focus on the pure algorithmic part of the implementation. The big advantage of these techniques is that they are not based on a complex system architecture. Therefore these solutions can be easily plugged into whatever kind of recommender system designed following some algorithm-independent approach [13].

The **model based** solutions move a step forward by creating a complete pattern of recommender system. Each model defines its item's *surrogates*, the profile generation and maintenance. The algorithms used then for matching purposes might be analytically selected, based on the desired system's behavior.

A different overview differences each approach according decisions taken when designing item *surrogates* are mainly guided by the approach selected to estimate the utility of a given item A for a particular user U. There are two main branches for this overview, on one hand based on the social properties of networks, such as the *collaborative filtering* [3], on the other hand relied in the user interaction and preferences, like *content-based filtering* [4]. The proposal described in [1] studies the possibility of combining both techniques, referred as hybrid recommendation systems, obtaining finer recommendations from better suited user profiles. I found further complete the solution explained in [6] and resumed next.

- **Content-based systems:** item *surrogates* will be composed of attributes that characterize their information content.
- **Collaborative systems:** item *surrogates* are reduced to their minimum expression, and their utility estimation is more a matter of statistical or probabilistic prediction.
- **Context-based systems:** item *surrogates* are composed of contextual information.
- **Hybrid systems:** using combination of all of the above methods.

2.5. Conclusion

This review shows the variety of decisions to make when planning a recommender system, offering a complete summary that eases the decision making process upon analysis' phase. Some decisions visualized after this analysis state that the user profile employed in this recommender could be based on history based generation, due to the monitoring capabilities of the web interface itself. The *user profile* will be refreshed when new information is retrieved from interaction, therefore *user profile* is continuously evolving. The only relevant feedback taken into account for this purpose is the purely implicit. It is retrieved optimally by the case-designed web interface. The most suited solution found for this purpose relies on logic-recommenders as mathematical model and data domain representation, while its features as recommendation model still could be more precisely estimated after next section, where music recommenders are introduced.

3. Music recommender systems

3.1. Overview

There exist a variety of web systems ready to help users discover new music, some are commercial applications, some are open source projects easy to inspect, therefore providing useful information about their model design. Commercial systems are completely closed to users, no reviews detailed in their documentation or internal logic explained at all. Some documented users could infer the model of these services, but it is impossible to get detailed information about the recommendation algorithm or user profile concept implemented in this systems.

Most of *commercial systems* often implement a complex recommender structure, some examples are Last.fm, Grooveshark or Spotify. All of them incorporate a music recommendation algorithm as an important part of their working. This algorithm is an information-filtering system itself, which plugged into musical systems, tends to sharp the music collection presented according the user's preferences. Some of the most important (in terms of popularity) music services are Last.fm[7], Pandora[8], Spotify[9], Magnatune[10], but also implement recommender algorithms lots of Internet sites as Apple music store[11], (ITunes is the most popular according [5]), or the Amazon e-Commerce website[12].

There exist pure *music recommenders* like Emergent-music[13] inside commercial world and for inside non-commercial world as iRate[14]. Despite these systems are not large complex communities as those mentioned above, they successfully fulfill recommendation actions. It ought to be considered that collaborative approaches strongly depend on the number of regular users the system is managing. I understand a regular user as someone interacting frequently with the system; otherwise, a latent user will not be useful for collaborative purposes. Some of them are reviewed next while those selected to be used for this purpose, upon being deeper studied; are listed in section 6.2. First Iteration

[7] Last.fm website: www.last.fm.com
[8] Pandora system: website: www.pandora.com
[9] Spotify desktop system: www.spotify.com
[10] Magnatune radio website: magnatune.com
[11] Apple Itunes Store: www.apple.com/itunes/what-is/store.html
[12] Amazon website: www.amazon.com
[13] www.emergentmusic.com
[14] irate.sourceforge.net

3.2. Approaches to music recommendation

The *collaborative filtering*, whose recommendation heuristics depend on the rankings established by the system users [6], sets out some important efficacy problems. It can happen that some song or some artist isn't ranked by any user, this artist or song won't be presented as a recommendation itself [7]. This mood clearly decreases the number of songs or artists available for recommendation, just because the items tend to be ranked increasingly and the most commercial music will be on the top of ranks due to popularity not to real user likes. The social features of the collaborative filtering converts the recommendation technique into a kind of filter which depends on social indicators not really in the user musical preferences, inviting the user to try something because some people liked it before, forgetting a little bit about the current hearing intentions.

The *content-based filtering* selects items based on the correlation between the content of the items and the user's preferences as opposed to a collaborative filtering system that chooses items based on the correlation between people with similar preferences [6]. Music content is often classified with some metadata tags, as artist name, genre, year, release country, and so, but sometimes audio features are also analyzed like melody, harmony, bass or tempo. These features are conveyed as *surrogates* about content. The available music not matching any feature of some played song or artist, will never be presented to the user as a recommendation, this causes the isolation of many music styles unknown for users [7]. So there's not too much chance of discovering some new music using this kind of heuristics if no extensions are performed.

The most successful music recommendation systems combine diverse features from each model [6], generating quite different approaches [23]. Many models feature new specific algorithms to achieve the same objective: Provide the user with new and interesting music information.

3.3. Commercial systems

3.1.1. LAST.FM

Last.fm website[15] is one of the most outstanding music recommenders out there. It clearly illustrates the concept of *collaborative filtering* recommender system. Users access recommendations by connecting to a web-based music streaming service. The tracks played on that stream are the recommended items. Like while listening to the "random" broadcast, users can tell the system whether they find the item being broadcasted interesting or just plainly ban the author of the item being broadcasted.

There are two kinds of recommendations streams: one for subscribers and another for non-subscriber. Depending on the user being a subscriber the recommendation algorithm precision varies. In the cases of non-subscriber users, the items broadcasted are selected according to a group of user profiles that are found to be similar. Subscribers can access a music stream whose contents are governed only by their user profile. It is then expected that the items on that personalized stream match more closely user preferences.

Audioscrobbler.com is an open source project[16] that acts as a *data harvester* for Last.fm web service. It uses and requires functionality of a quite complex and expensive infrastructure. This seems to be mostly paid through a donation system, where users are expected to donate the amount they feel the system deserves. Users that donate money become subscribers accessing enhanced services.

In order to build up the user profile, the Las.fm system has implemented three different approaches:

- User adding explicitly items (artists) to their profiles through Last.fm web interface.
- Get the AudioScrobbler.com plug-in, available for a wide range of media players, which records which tracks are played. Once a certain number of *playback events* have been recorded, a report is sent automatically to the Last.fm servers. This information is integrated together with other previous statements in the user profile.
- User can connect to Last.fm Radio, consuming a stream of music over which features a significative proportion of not very popular artists. Users, through a set of web controls in the website, can tell the system whether they approve or ban the artist whose work is being played at the moment. This feedback is also integrated into the user profile.

Last.fm has chosen a very simple approach to elaborate recommendations, even if they are good some limitations can be observed that are related to the probabilistic-nature of the recommender algorithm. One main limitation is that the system requires a huge number of observations in order to get good estimates of the conditional probabilities.

[15] www.last.fm
[16] www.audioscrobbler.net

3.1.2. FLYFI (Emergent music)

Emergent Music[17], at starting point presents the user with a list of top downloaded tracks and top listened tracks for the current week. This certainly suggests some collaborative filtering inside their recommender engine. The user can create playlists of songs which are saved automatically inside the user profile. Users are allowed to either download published tracks or listen to them through its streaming service. However, there is not always the option to do so: artists decide whether to make or not publicly available their works.

The playlist creation feature acts as an user activity sniffer, creating relations between songs. These relations and the songs included help the recommender to build the user profile. Recommended items are ordered by its expected affinity with user's taste. Besides that, users can also perform simple searches on the recommended items, specifying several keywords. Feedback on recommendations is given by explicitly rating of presented items. The interface offered for this task is quite simple. The problem comes with the number of recommended items which may be greater than one hundred, which implies a hard task for the user to give each item individual feedback.

Emergent Music is a Music Recommender exclusively based on collaborative filtering algorithms and techniques. It also provides a desktop application called Goombah. It has a more complete interface than the web system, interacts with their database to provide the user with recommendations and music associated to the played song. Other feature is a partnership-like playlist scrobbler[18] in association with ITunes. This software must be installed locally and acts as a boosting element for the recommendation engine. It creates relations among the tracks listed inside the current ITunes playlist and between the user profile and these tracks, which are loaded from the user's music collection.

[17] Flyfi web site: www.emergentmusic.com
[18] From Last.fm scrobbler engine: www.audioscrobbler.net

3.2. Non-commercial music recommenders

3.2.1. JAMENDO

Jamendo is an online community[19] of free, legal and unlimited music published with Creative-Commons licenses. All the music published on the Jamendo site can be used free for personal use. The Creative Commons license allows the owner of the music to retain some rights, while giving users the ability to download and listen to this music freely. Commercial rights are applied to each musical piece separately and are handled through the Jamendo site. This allows the site to offer free music to its users yet allows publishers to earn some income from the commercial rights to their works.

The site also features a group of selections, Radio Stations, and Playlists created by users. The site also features social networking aspects such as user profiles, user friends, community forums and inter user-messaging. Songs can be streamed or downloaded, depending on the copyright laws ruling in the country where the download is sent.

The music stored in Jamendo database is cataloged by artist, album and by tagging options. Content-based filtering is present in the recommendation engine as well as a collaborative solution that models the interaction between users. There exist user clustering classifiers that can be composed using some of the many interaction possibilities a user can have with other system elements. Some features as groups of friends or the internal messaging among them are quite useful for a collaborative approach due to its semantic contribution by linking their musical preferences with the common features defining their profiles.

The system provides a web service, which can be used by developers for example for adding free music to their web sites. It is provided an api documentation[20] where it's explained clearly what to do to interact with this web service.

Jamendo is an incipient music recommender with constant growing and increasing popularity nowadays. Now it has more than 10.000 albums available for streaming or download. They provide help for new groups to promote their work offering flexible licensing features. It's a really interesting web site but still with an immature music catalog.

3.2.2. iRATE RADIO

iRATE[21] is an Open Source project, whose purpose is to help artists to publish their works. Conceptually is a file-sharing application, where artists publish their works on the iRATE servers by adding links to the files they wish to share. iRATE distributes this music by presenting network users with recommendations.

The recommendation engine is mainly made up by a collaborative filtering system strongly influenced by user feedback. The system creates correlations between user profiles

[19] Jamendo website: www.jamendo.com
[20] Api documentation for Jamendo web service: developer.jamendo.com/en/wiki/MusiclistApi_draft
[21] iRATE Project: irate.sourceforge.net

and their track ranking to achieve the clustering all over the users.

iRATE developers emphasize that the system does not intend to become a smart P2P network. This is further enforced by making sure that the only music made available is licensed under the any of the Creative Commons licensing patterns. Users are required to rate explicitly the songs the system presents them. This is achieved by getting the users to install a Java-based application, which downloads the music published on the system servers. As soon as the client application downloads a song, it is played back at the user. The user has the opportunity to rate it using the client application user interface. Songs rated with the minimum score are immediately stopped by the system, starting the playback of the following song if exist.

3.2.3. FOAFING THE MUSIC

Foafing the music[22] is an open project of the Pompeu Fabra University sited in Catalunya, Spain. The system uses the Friend of a Friend (FOAF) and Rich Site Summary (RSS) vocabularies for recommending music to users. The FOAF vocabulary contains terms for describing personal information like name, nick, mailbox, interest, images, membership in groups, organization, etc. This vocabulary helps the system to build consistent user profiles which include relations among users.

The recommendations are built by getting information about user preferences from the FOAF profile, then, it checks using a music repository whether the interest is about music artist and selects several similar to the items found. In order to collect these similar artists, the system creators have developed a focused web crawler that searches the relations between artists. After collecting many similar artists, the system scores them depending on the number of playbacks of each song.

The system's most interesting feature is that upon the artist of interest according the user profile are selected; the system filters music-related information from Rss feeds. From this filtering process the system will retrieve new music releases, download or audio streaming links from mp3-blogs or Podcast sessions and audio playlist generated based on audio similarity.

Music Related News

An extra feature this system includes is the music related news which are all about the musical interest inferred from the user's FOAF profile. They make use of a very interesting service called Pubsub[23] whose purpose is to maintain up to date a big information index. Pubsub collects news from over 13 million weblogs and around 50.000 newsgroups, and adverts the user if some new content matching his search terms have been published.

Upon the news are collected, the FOAF system uses the TF/IDF algorithm to score the news documents and present them to the user ordered by relevance, as explained in [12].

[22] FOAFing the music Project: foafing-the-music.iua.upf.edu
[23] Pubsub Website: www.pubsub.com

4. Problem definition

Aim

This project aims to create a free web-based music recommendation system able to estimate the user's musical preferences and elaborate recommendations of several musical elements according to these preferences.

Information sources

The system intends to use various online music services, through which obtain listings of groups and artists presented to the user. The music collections retrieved from music services act as a browsing environment to let the user navigate through music. Each music service provides several features, some common to all, others are particular features. The system aims to combine the traditional functions of various musical information providers, to get more results and more information to offer.

User interaction

The user interaction is done through a web interface accessible from any platform with a web browser. This interface provides great opportunities for interaction enabling continuous navigation through thousands of albums and artists. It is designed to allow the user easy and intuitive interaction, it should be mentioned that the longer the interaction is, the more complete information about user´s likes is stored, therefore better recommendations generated. This recommender system is basically a software element that studies the user's browsing patterns and then decides what to present next.

System features

In order to provide the system with a complete musical collection, several music web services have been reviewed. Unfortunately, the lack of a relational music database limits somehow the freedom for managing music data in my own way. Therefore the music catalogs are loaded dynamically from these music services when the user interaction requires them. Upon this data is loaded, the system extracts the significant information about the musical items (item *surrogates*) to evaluate what kind of music the user is interacting with. By monitoring this interaction the system is able to build a user profile, which is not understood as a constant definition of user's preferences, instead it's conceptualized as an adaptive changing pattern. In this way, the system is able to store a historic of user interaction as a long-term user-system relationship but still reacts more sensitively to recent occurred events, preserving the system from over valuating the most frequented items, storing also mid-term interaction memory.

Improvements

It has been observed that most music recommenders rely on *collaborative filtering* techniques to support sometimes or to boost others the recommender system functionality. The nature of this filtering slightly diverges from the *pure* concept of recommendation, which is strictly based on the current user's preferences.

It has been proved in [25] that *collaborative filtering* provides good recommendations to users with no previous knowledge about user likes (explicit). The fact inspiring this project's aim is to base recommendation explicitly in implicit information retrieved from user actions.

The problems commented in section 6.4 have been probably solved by current commercial systems due to experience obtained during its time *online*. It's believed so because these systems provide good recommendation results to users as well as economic benefits to founders (in the opposite case they won't be online). This project offers a content-based context-based recommender, able to provide *new* musical content, without being *influenced* by any collaborative-like procedure. Problems related to the content-based model described in section 3.2 have been solved using custom designed algorithms (section 6.3 and section 6.4) as commented in section 6.4.

5. Methodology

The selected methodology used for developing this work is the Rapid Application Development explained in [10]. Its characteristics fit very well to the needs identified after the planning of project execution. These characteristics are:

- Iterative
- Based on goals and use cases
- Using GUI tools, CMS, etc.
- Periodic testing system
- Track Changes

The nature of the problem makes necessary several iterations over a changing pattern. This model will be continuously improved, as we proceed through the knowledge of the data with which it works, more reliable will be its behavior. The aim of this iterative process is to refine the relationship with the chosen web components whose reaction is not known a priori. The iterations are designed to improve the system performance about web components.

Iteration's steps:

1. Determine objectives, alternatives, and triggers for iteration.
2. Evaluate alternatives, identify and solve problems.
3. Develop prototypes and verify the results of previous design.
4. Specify objectives for the next iteration.

6. Development

6.1. Summary

The project development from the first analysis of the tools necessary to implementation and final testing has been carried out using the method of rapid application design. The most outstanding feature of this methodology relays on its iterative nature. The overall process has been divided into several stages of development. Each stage is determined by previous targets and final conclusions setting out the objectives of the next iteration. Each stage includes activities related to analysis, design, implementation of prototypes and their late testing. The following summarizes the stages that emerged during the project planning and the activities belonging to each one.

6.1.1. First Iteration

1. Objectives:
   - Estimate the potential of music services available.
   - Determine a developing framework if convenient.
   - Check the characteristics of data from music services.
2. Analysis
   - Some Web services available and free music
   - Functionality offered by these.
   - Estimate advantages and disadvantages of using a CMS for development.
3. Design
   - Methods to access music services.
   - Elements of communication with music services.
   - Web structure for elements of communication.
4. Prototype requirements
   - Web platform that enables communication with the music services, to ensure data collection and allow further analysis.
   - Data type that encapsulates all types of data obtained from music services
5. Conclusions
   - Deeper knowledge about available music data
   - Election of a framework
   - Acknowledge limitations in the functions provided by the API of web services

6.1.2. Second Iteration

1. Objectives:
- Establish a methodology for recommendation.
- Start building the web system according the selected framework.

2. Analysis
- Definition of user interaction with the system.
- Algorithms for recommendation

3. Design
- Consolidation of data from several music services.
- First version of user interface.

4. Prototype requirements
- Prototype testing for check behavior
- Simulation of final web system and incorporate the prototype of the recommender algorithm.

5. Conclusions
- The recommender system does not satisfy the forecasted objectives.

6.1.3. Third Iteration

1. Objectives:
- Finding an alternative to previous algorithm.
- Implementing User Management
- Bugs and exception handling in web system

2. Analysis
- Feasibility of developing new recommender algorithm adapted to the problem.
- Algorithm domain overview.
- Study final interface.
- Solutions for user's management.

3. Design
- Technical design of an alternative recommendation algorithm
- Connecting the access control module to the web system

4. Prototype requirements
- Expanded web-based system implementing the user access control.
- Data persistence system running.

5. Conclusions
- The project's objectives are covered.
- System expansion options.

6.2. First Iteration

Objectives

Before starting the development of the application it is verified that the necessary elements for building up of the system are available. The system needs a great amount of musical information that ought to be presented to the user in a visually appealing way. Dealing with musical artists or groups and their albums, their description should be accompanied with some picture. This visual information improves interface intuitiveness easing the interaction with the system, while making it more attractive. In this phase, the tools designed to verify the data will be ready to run. Besides the visual aspect, musical content offered should be available for listening. At least a small sample of the song to guide the user on the "type" music he's actually inspecting. It would be checked whether music services actually provide listening options, the number of available songs for streaming, use limitations and so.

Analysis of music web services

Several music web services have been reviewed, focusing on what functions do they provide to developers in order to mainly determine the feasibility of the project. Among the reviewed music services there are Emusic[24], Yahoo music[25], Soundcloud[26], Discogs[27], Rhapsody[28], Musicbrainz[29], Last.fm[30] and Play.me[31]. The following table [fig 1], summarizes the characteristics of web services and the methods provided for free, as explained in the API documentation of each one. In this classification, are included only some of its features, those essential for the system regular running. These features are related to the classification of the artists and their works, genre, ability to make searches by albums, artists or songs, playing tracks, etc.

[24] www.emusic.com
[25] new.music.yahoo.com
[26] soundcloud.com
[27] www.discogs.com
[28] www.rhapsody.com
[29] musicbrainz.org
[30] www.last.fm
[31] www.playme.com

	Response format	Album search	Album related	Album genre	Artist search	Artist related	Track search	Track genre	Label search	Song Streaming	Limit
Emusic	multiple	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes, private player	No
Yahoo music	multiple	YQL {music.artist.id, music.artist.popular, music.artist.search, music.artist.similar, music.release.artist,music.release.id, music.release.popular, music.release.search, music.track.id, music.track.popular, music.track.search, music.video.category, music.video.id, music.video.popular, music.video.search, music.video.similar}								No	5000
Music-brainz	Xml	Yes	Release groups	No	Yes	No	Yes	No	Not always	No	No
Last.fm	Multiple	Yes	Yes	Yes, tags	Yes, Artist genre	Yes	Yes	Yes, tags	No	No	No
Sound-cloud	multiple	No	No	No	No, by track	No	Yes	Yes	No		No
Discogs	Xml, header gzip	Yes, by release	No	Yes	Yes	No	No, by release	No	Yes	No	5000
Rhapsody	Xml	Not provided									
Play.me	Multiple	Yes	No	No	Yes	Yes	Yes	Yes	No	Yes	No

fig 2 Table briefing the most outstanding features searched and the related support of each web service.

A conclusion derived from this review is the variability in the functions provided by each service. The difference between web services themselves is evident when focusing on storage facilities, services range from a single pool of songs and their metadata like Soundcloud, to large relational databases as Musicbrainz web service.

After evaluating the web services and their methods, were further studied access platforms (API) of some web services that have been considered the most profitable for this purpose. The selected web services are briefly commented next:

Musicbrainz

Free web service. It is a huge relational database of music, which can be accessed through a XML-based web service. Contains the greatest amount of musical information, the data amount is impressive, it contains several millions of songs, thousands of artists and albums is awesome. The problem with its use in this project is mainly the lack of graphic content, despite a link that refers to the corresponding item on Amazon, no links to listen or download tracks are available.

It's a big music library that provides textual information to find new artists or new relationships between them and their albums, but because of its size, there are many stored items that cannot be presented to the user for not including much needed information as links to pre-visualizations or album art images. Due to this lack of multimedia information about music elements, this service is not going to be used for this project.

Last.fm

Last.fm provides developers with a powerful web service with a huge and comprehensive music catalog. This service provides many details about groups or artists and their albums, as well as images in various sizes for artist and album art. It also includes a tagging system in which users can add personal tags that can be used to classify music privately as a way to create personal collections. Many tags are applied to each artist, album or song depending on their popularity. Tags help to extract the musical genre of the album or song, among other things, being a serious drawback that these tags are not present for every artist, album or track. The complete web service and musical catalog makes the last.fm web service to be one of the most suitable for the purpose of this project, but also has some limitations: it doesn't allow listening or downloading songs. The API does not provide solutions for previewing or downloading a track, listening is only available through the web site, and only by payment account.

Play.me

Tests using the Play.me Web Service determined that not all items announced on its API website have real support in the web service.

It has been implemented a software layer or wrapper to access the web service API, intending to evaluate the usefulness of the service for this purpose. The information provided inside the API documentation and returned-data definitions are not completely accurate. At the time of representing the information graphically in the browser, the images of artists and albums are crucial. With them, the platform providing information to the recommender algorithm has a rich visual interface which makes it more entertaining than a pure text-based web page. The images about musical items are represented as links to the content, but often do not charge due to broken link or, unhopefully, due to internal server errors. Play.me web service includes a very interesting field inside music items, this is the genre, very useful in order to catalog content and thus be able to recommend music based on musical genre. Other interesting feature is the possibility of listening a short 30 second sample of each song.

The most serious drawback found for Play.me is the musical catalog, even if it is big, is not comparable with the one owned by Last.fm or by Musicbrainz.

Development-framework analysis

A good study of the features offered by different development frameworks may save time and be decisive for achieving a robust and efficient final result. The content management systems or CMS, provide effective solutions to manage users, exchanging messages between them or sending electronic mail. Also facilitate the control of access to webpage elements with roles assigned to different users. Another function of CMS is to facilitate the maintenance of a web site, with methods to update content such as news or articles by performing simple actions.

Some time has been spent analyzing several open source CMS, including Magnolia[32], AtLeap 1.0[33] and Pligg[34] which includes social networking features. All of them provide plenty of facilities for the maintenance of web sites. However, what is sought for this project is to simplify the management of potential users of the application and make use of a framework for safe and easily maintain their profiles. Most of these systems are designed to manage web sites of news or articles, short stories and frequently updated information, providing further support for user authentication systems with different levels of access.

This system needs to manage users, but the dynamic characteristics of the application, do not fit with the facilities offered by these CMS. Communication between users in this case, is practically absent. The system aims to focus on the user, not in the relationships between users. The system can be easily expanded to extend the communication between users, but this feature remains as an outline of this work intention.

Design of communication element with music services

The system needs information from music services to provide interaction, now it is defined how to allow communication with the web services of each provider. Ideally, a single Web service, providing all the information necessary for the application would simplify the system's web architecture and also a better performance could be achieved. However, no free music service providing a large multimedia music collection in order to complete the desired music objects.

For this reason, it is necessary to implement several components that allow communication with the selected music services. The information provided by services: the methods available, the structure of metadata and answer format, have different characteristics depending on each service. Play.me offers opportunity to listen to tracks, a thirty seconds simple. Last.fm classifies information using labels, but the lack of genre classification inside the Musicbrainz web service reduces drastically its usefulness for this purpose.

Communication with music systems is done through their API. The interaction is performed using http requests which are sent to the each web service.

[32] www.magnolia-cms.com
[33] atleap.dev.java.net
[34] www.pligg.com

Communication with Last.fm web service

In order to make use of the Last.fm database, they provide API documentation[35] where to find information related to methods or response formats. It's possible to find previous implementations of wrappers written in several programming languages. There exist some Java bindings written by Janni Kovacs which are BSD-licensed[36] and are available for development purposes. It is a project hosted on Google code[37]. Not all the features implemented in this wrapper have been used, only the related to musical content retrieval. In order to establish communication between the html test page and this service, a middle Java servlet[38] has been implemented to receive Ajax[39] requests from the browser and bridge them directly to Last.fm web service. The serialized data sent back by the web service comes formatted in json[40]. This object definition language allows perfectly the object notation as the time its structure eases quite much the parsing process.

Communication with Musicbrainz web service

Musicbrainz xml web service[41] is a huge service. It's being continuously growing due partially to its feature that allows user to easily send metadata about music releases. In order to communicate with this service it's possible to use some useful java code also available for free use. This wrapper also hosted on Google code[42], implements a complex set of functions that allows access to some interesting tools available in this web service. Only music information retrieval functions have been tested. The xml data is parsed and converted into java objects, this utility has been employed for checking data integrity, response time and other issues related to communication. Unfortunately Musicbrainz has no truly usefulness for this purpose.

Communication with Play.me web service

Play.me web service does not have any API wrapper coded previously by the community. A Java API has been created to provide communication with the API of this Play.me[43]. As the others, the service receives http requests and sends back responses formatted in various data-types as shown in [fig 1]. After having a look to the methods provided in the API documentation, the related to music retrieval have been implemented and tested. It is necessary to check whether the data sent back by the web service meets the specifications inside the API documentation.

[35] Last.fm api documentation: www.lastfm.es/api/intro
[36] creativecommons.org/licenses/BSD
[37] Last.fm java bindings: code.google.com/p/lastfm-java
[38] Servlet technology: www.oracle.com/technetwork/java/index-jsp-135475.html
[39] www.ajax.org
[40] www.json.org
[41] Musicbrainz xml web service: musicbrainz.org/doc/XMLWebService
[42] Musicbrainz java code: sourceforge.net/projects/javamusicbrainz
[43] Play.me API: lab.playme.com/api_overview

The element of communication uses these software components that connect the system with web services. Upon receiving a request from the web interface, the communication element prepares the requests for each music service. The responses from each music service are analyzed by extracting the necessary information from each service response, and then generating a full data object with which the web system can perform its activity. The following diagram represents the structure of access to music services and communication with the web component.

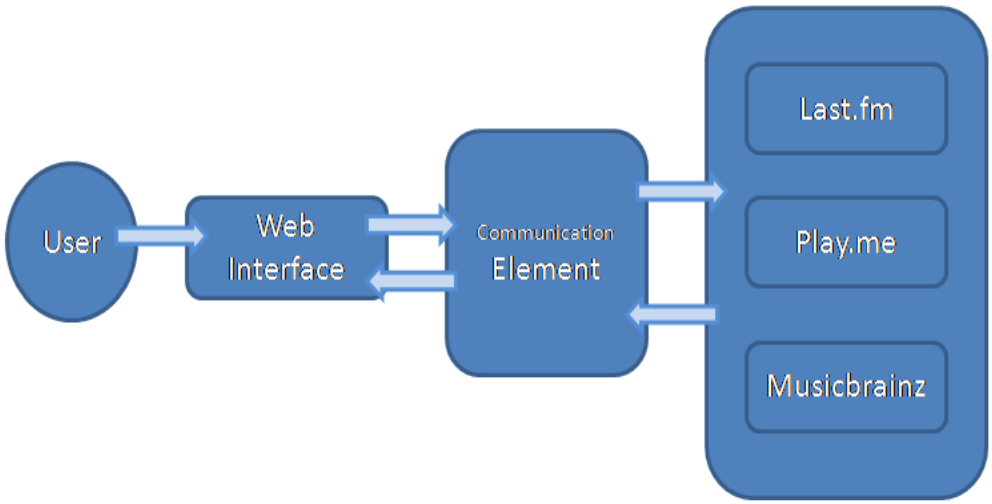

fig 3 First system's prototype structure

Prototype's requisites

The implementation of the prototypes to allow communication with music services makes possible accurate verification of specifications detailed in the respective API documentations. Many methods promised certain features that after testing, it was found that were not covered or function behavior was not the expected. Also allows studying the actual structure of the data returned on each call:

A call to the Web service Last.fm, which are requested albums belonging to a particular artist, returns a collection of elements, with a particular structure. For example:

```
Album { id, albumname, artistname, genre, { tracks } }
```

It happens that the structure is not complete, receiving the call basic information about the item in question in the form:

```
Album { id, albumname, artistname, *, { * } }
```

To complete the structure, it is necessary to submit a second query, which calls for the metadata associated with the ID of an album or track list identifier associated with an album. Once the requests, you can have the necessary information.

This event converts an ideal request to the web service, which is associated with a call for artist albums, in an initial request, plus, as many requests as albums associated to the artist being queried. The consequences are clearly negative for optimum application performance and the proposed solutions are many:

- A selective analysis of received information, minimizing the number of consecutive requests to the web service.
- User data from each web service when it best suits.
- Eliminate the need for extra information such as gender, the labels associated with an item, previews, etc. reducing the system's utility.

The disparity between the music catalogs of selected services poses another question when generating the data type the system will work with. The problem is that the catalog of Musicbrainz, probably the most comprehensive of those selected, provides no images or multimedia information related to the items stored. It is needed to provide identifiers of the elements stored in other services such as Amazon, which provide images for artists or album covers. Last.fm, with an equally vast and varied catalog, still offers pictures and album arts, also offers a tagging system that classifies artists, albums and tracks, but offers no previews like Play.me service. However, Play.me service, which also offers images (although with broken links to images of artists) and lists artists and tracks, also features tracks pre-visualization has an immature catalog.

The most complete catalog has no multimedia information, the best overall, is Last.fm, but does not allow pre-visualization, while Play.me can be used to play samples of songs, but has a smaller catalog. Using the current strategy to build up the musical objects and affected by the disparity of music catalogs would retrieve incomplete musical objects. Some would be loaded from Last.fm, but could not be listened, others from Play.me, instead could be streamed but maybe images of album arts are not loaded. Musicbrainz service can only be used to check relationships between groups that should be also stored in some of the other two services, preferably both, to avoid object incompleteness.

As a result of tests with the prototype, the use of Musicbrainz music service has been rejected. It's a music database with information in text format only, which is useful as a source of musical knowledge, but not useful for the system being developed which is based mainly on multimedia content. To supplement the information provided by Last.fm, which does not include previews, the famous video site, YouTube, is proposed as an interesting alternative, due to the vast number of music videos growing each day.

Conclusions

After studying music services, I delved into the techniques used to relate the music, mostly used by Musicbrainz and Pandora. These services make mention of *audio fingerprint.*

The audio fingerprint[44] is a summary of the acoustic characteristics deterministically

[44] wiki.musicbrainz.org/AudioFingerprint

generated from the audio signal of an element. It can be used to locate similar items focusing on acoustic characteristics within a database of music, or to recognize a particular piece of audio using these characteristics recorded in the fingerprint. None of the recommender systems found contain the fingerprint of every song, neither allow the track search using fingerprint. Last.fm is currently collecting fingerprint of their songs and getting some of the users in an effort to improve its recommendation system.

It is possible to develop a configurable system where the user can ask for, or be recommended when he desires, with elements that contain similar acoustic characteristics to the songs he listens to.

In order to develop some commercial music recommendation system, the designer should consider carefully the proper organization of the music database relating properly the artists and genres with releases and publications dates. Including fingerprint indexing coverage for all songs would provide complete musical relationships among songs. With this information properly indexed would provide the potential of providing a robust and complete music recommender sensitive to music features, not only to relations based on music metadata.

6.3. Second Iteration

Objectives

To establish a methodology of recommendation it should be specified the scope of interaction we want to offer. Once the limitations of music services that will be used are known, it is possible to establish with certainty a clear pattern of response to user actions, these actions would be encoded, extracting *surrogates* from musical objects. These *surrogates* represent the data used by recommendation engine. The system being implemented should be able to recognize user's music preferences by analyzing its interaction with the system. No patters or likes are determined a priori, but the content shown is adapted according the pattern of interaction constantly recovered. This stage starts from the first sketch of the web interface design. The component allowing communication with web services is being to be implemented as a data collector for the web interface. Upon this pieces are working together is easier to determine the recommender domain.

Defining user interaction

The user interaction with the system is done through a dynamic web interface. For the interface development, all the possibilities the user might need to manage music content have been identified. From all this interactions, the system extracts information used to infer the user preferences. The following table resumes all the available interactions:

Action	System response
Insert search text	Loads results related the introduced term
Artist select	Loads albums related to the given artist as well as similar artists to provide furher interacion options
Album select	Loads the tracks and the playback options they have as well as albums related to the given for further interaction options
Play track	Generates a web player playing the selected song as well as some similar tracks in terms of metadata
Log-in, log-out actions	Loads or saves user profile, and prepares the interface according user’s preferences
Stop playback	Not ranking the item related. The system is not feed upon negative events
Rate artist, album or track	Increases the vote for the given element and the related information like musical genre

fig 4 User interaction description

Select a recommender model

The recommender system analysis offers multiple strategies for developing internal heuristics. As reviewed in previous chapters collaborative filtering is commonly included in most of the recommenders nowadays, mainly based on user ratings to recommend artists, albums or tracks. This strategy is useful if the aim is to create clusters of similar users who share tastes and provide critical reviews for the rankings of artists, albums or tracks, through explicit qualification of one of the elements. Clustering techniques to group common features are commented in [14].

In this way, the system can recommend to a user A, a record that someone else chose (user B), while listening a track which is directly related to both. In order to achieve this feature, most of the reviewed solutions rely on relation matrices, where the objects are related to user profiles, and these relations are as well rated to assess their importance. Keeping rankings of music elements and relationships among groups of users is out of this project's aim. It is convenient to remark its importance inside recommenders' heuristics but not will be studied deeper in this paper.

For this project, the actual collaborative filtering does not exist. There is no ranking of songs, albums or top artists. One of the reasons why the collaborative approach has not been chosen, is about this system does not intend to use a specific music relational database. Instead it is loading information "on-the-flow" directly from music services and re-configuring dynamically the web interface. Other more important reason is the main goal of this project. It is not about developing other collaborative recommendation system, but to develop a system that is able to infer the current user preferences and able to elaborate a response according those preferences through the use of free music services.

Recommender algorithm's review

1. k-Nearest Neighbors (kNN)

The k-nearest neighbor[45] full explained in [26], classifies an unknown object O with the most occurred label among k-nearest neighbors. A neighbor is considered nearest if it has the smallest distance, in the Euclidian sense (angle cosine), in feature space. For $k = 1$, the selected label belongs to its closest neighbor in the learning set. The k-nearest neighbor method is very intuitive, and for this reason broadly used.

When used for regression, NN labels are real numbers and the task is then, to interpolate the numeric label for O the object pending of classification.

The discrimination function implemented by this classifier will in general be a rough, piecewise linear function since it is influenced by each object available in the learning set. A disadvantage of this method is its large complexity and power requirement, since for classifying an object its distance to nearest k objects in the learning set has to be calculated.

[45] www.lkozma.net/knn2.pdf

This algorithm, when interpolating class labels, has two parameters:

- *k* or number of neighbors to be calculated to infer the label.
- the kernel function to approximate the numeric relationship.

kNN-based recommendation algorithm

The learning set is the set of rated items appearing on

were are the item *surrogates* limited to real value features. In order to infer the utility for an unrated item , we compute the distance vector .

where L2 stands for the Euclidean distance (second order metric). This vector D is normalized
to lie in the [0, 1) interval according to the maximum distance

Once distances are normalized, we linearly interpolate the utility for item

where is the number of elements in . Note that since utility assignments are either 0 or 1, it is needed only to consider those objects that were assigned non-zero utility. The full recommendation algorithm pseudocode is shown next. In this version, parameter k is , the total number of rated items in Λ.

is the set of items eligible for recommendation, ordered according the predicted item utility, and is the set of items that will be presented to the user. It requires a training algorithm which should be designed according the system behavior.

K-nearest neighbors algorithm

Input Λ: rated items set, unrated items set, number of items to recommend

Drawbacks

Despite the k-NN algorithm is conceptually *simple to apply* its computational cost for predicting over a set of unrated items given a set of rated items is which is quite expensive in terms of efficiency.

Application to this context

The constraint of numerically representing the item *surrogates* and the user profile features highly increases the uncertainty about its fitting for solving the current problem. The fact of representing numerically item *surrogates* like artist name, or musical genre entails some loose of significance as the time of being a complex representation issue by itself.

It could be possible to place a musical items into a n-dimensional space, by using audio signal features, but as much as known until this point, the only way to extract item's features using the data collected from music services is by reading artist features, like genre and in some of them the release date and tag the music object with textual information.

2. Bayesian Network

A Bayesian network[46] is a *graphical tool* that allows to build models representing processes with inherent uncertainty, as explained in [15]. This model takes the form of a joint probability distribution[47]. Besides this modeling ability, they also allow to study how changes in the uncertainty of one of those variables affect the others. When a change in uncertainty of variable X implies a change of uncertainty in variable Y, it is said that there is a *dependence relation* between X and Y. It is also said that X is the *parent* of Y.

Bayesian networks are made up by two elements:

- A set of variables and a set of edges conforming a *directed-acyclic graph*.
- To each variable with parents there is attached a probability table .

The first thing to have in mind when proposing a Bayesian network model is that its purpose is to give estimates of probabilities for events that are not directly observable. The first task is then to identify them. An unobservable event could be represented as the implicit rating a user gives to some item which still remains unrated. In order to follow the Bayesian model, these events must be packed as mutually exclusive events, for example:

When selecting some track for reproduction, its valuable *surrogates*, represented in this problem as track's artist and track's musical genre tend to be rated positively if the song is reproduced or could be rated negatively or ignored in the case the user skips some track that was being played. Assessing this relations between the user actions and changes produced on other information variables, it is possible to define the probability function for the hypothesis variable. These information variables can be called *hypothesis variables*. As said before, these are the variables composing the *surrogates* of items or It is assumed that the information variables are mutually independent. So some action affecting to *rock* genre, will not affect the *classical* genre, and so for the artist, changes over *The Beatles* group would not directly affect other *pop music* groups.

The probability function for a given hypothesis variable can be computed as follows:

where α is a normalization constant. The characterization of the probability distribution can be arbitrary decided following desired pattern designs. For giving an example in which all the possible outcomes are assigned the same probability, the Laplace's principle of indifference[48].

[46] http://www.eng.tau.ac.il/~bengal/BN.pdf
[47] Joint Probability Distribution: www.stat.tamu.edu/~henrik/211S05/notes/chp5.pdf
[48] en.wikipedia.org/wiki/Principle_of_indifference

Network parameters estimation

Since the variables in are independent, estimating the parameters for this Bayesian network is just a problem of estimating the parameters that define the distribution . Very much like while deciding which distribution choose for , we chose to model these distributions as unimodal, normal distributions. A Gaussian-probability distribution is commonly used for probability issues.

where stands for a numeric variable belonging to . Note that since we are considered conditional probability distributions, in practice, we will have to estimate two sets of parameters , one for each of the possible outcomes of . The simplest approach to obtain an estimate of these is to choose the parameters so that they maximize the likelihood of the data. In this case, the data are the pairs of utility assignments . Next it is needed to select good estimates for . Many previous works propone *the sample-mean* and *sample-variance* to be good estimates of distribution parameters, having an hypothetical continuous training data.

Bayesian recommendation algorithm

Input Unrated items set,	posterior hypothesis distribution estimate number of items to recommend	: hypothesis

Application to this context

The real problem really has a continuous training data which are the selections of music items, which arrive continuously to the data collector from the web interface as a consequence to user interaction. The problem is that this number of infinite inputs must be determined while performing the calculation thus estimation of parameters is still

mandatory. Other important drawback for application on this problem is the poor capacity of numeric abstraction over systems' item *surrogates*, which have independent textual significance and cannot be numerically estimated.

3. Algorithms based on music data

As described in [16], is it possible to achieve good recommendations without the need of abstracting self-significance keywords into numerical estimators, with the corresponding loose of information in the process of transformation.

In this paper they explain their *content-based method*. Based on the content-based filtering approach, the purpose of the CB method [16] is to recommend the music objects that belong to the music groups the user is *recently* interested in. In order to capture the recent interests of the user, they analyze the latest transactions in the access history as follows:

Each transaction is assigned a different weight, where the latest transaction has the highest weight. Moreover, the music group containing more accessed music objects in a transaction has a higher weight than other groups in the same transaction. The weight of music group is computed as follows:

Where is the weight of transaction , is the number of latest transactions used for analysis, , is the number of music objects which belong to music group in transaction .

These weights will be recorded in a *preference table* for the user. After calculating the weight for each music group, the recommender system ranks all the music groups. The music group with a greater weight takes a higher priority of recommendation. To avoid recommending a large number of music objects to users, the recommender limits the number of music objects for being retrieved. According to the , different numbers of music objects from the music groups will be recommended.

The *STA Method* is based on the use statistics [16]. They define a long-term hot music group as the music group containing the higher number of music objects in the access histories of all users. Furthermore, it's also defined a short-term hot music group as the music group containing the most music objects in the latest five transactions in the access histories of all users. When the user chooses this recommendation method, the MRS recommends the latest N music objects (which have not been accessed by the user), half from the long-term hot music group and the other half from the short-term hot music group to the user.

4. First Algorithm proposal

The system needs to collect information about user interactions for the recommender. For this, the user interface is designed in such a way it allows to obtain data related to the clicked items.

To elaborate a context that defines user preferences, every user interaction is stored in a profile-related structure independent for each user. This structure contains all the user selections made during his session. The recommendation is made when the structure has *enough* information to provide a possible recommendation. To decide when the information stored is actually providing an adequate pattern, can be arbitrary decided depending on the desired behavior. By establishing this threshold to higher values the recommender engine disposes more data to generate recommendations therefore being more satisfactory in terms of musical coverage. As counterpoint, the user profile evolves more slowly in time. The opposite behavior could be obtained by selecting a lower threshold . Each recommendation is executed upon this threshold is reached and iteratively the input data-set to perform recommendation increases.

Let us represent the recommendation input-set for the th time the threshold is reached, and the set of stored item *surrogates*.

For its iteration the threshold is refreshed as follows:

The algorithm uses statistical techniques to determine the most selected artist over the data-set . If some artist is dominantly selected, the recommendation is made about this artist, making an album compilation composed by selecting some albums from this artist and some from its similar artist's albums. This artist is stored as some artist that could interest the user inside the user profile. If no dominant artist is found inside the data collected, then the recommendation will be based on musical genre. The statistics applied on genre data collection tend to retrieve the rate of the most "hit" genre, are explained following:

- If is over a 66% of hits, then it's believed that this genre is interesting for the user. Therefore the recommendation will be based on artist related to that musical genre.
- If is over a 33% of hits, then the rule inferred is that this genre is not a decisive element able to represent the interaction mood, thus, a set of related musical genres

to the given is retrieved. Let us represent the set as the set of genre tags related to . Then artists are retrieved for each musical genre inside .

and parameters can be configured in order to obtain a *wide* [fig 5] or *deep* [fig 6] recommendation set. Setting N large and M small the system would provide a result set rich in genres while discrete in the number of artists belonging to each genre, as a *wide* set. Setting N small and N large not so much genres are requested as the time more artist are retrieved for each one, as *deep* recommendation set. Figures show in green color the conceptual size of artists sets in from of the orange color representing the genre collection set.

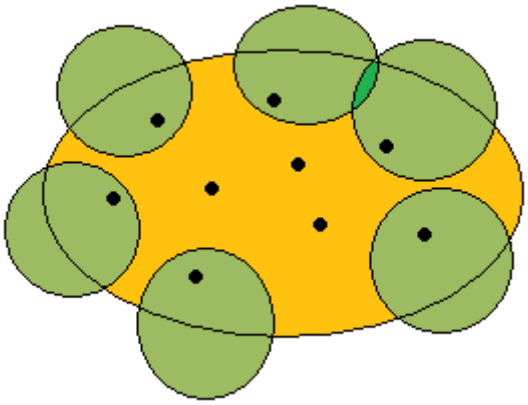

fig 5. Wide data-set representation

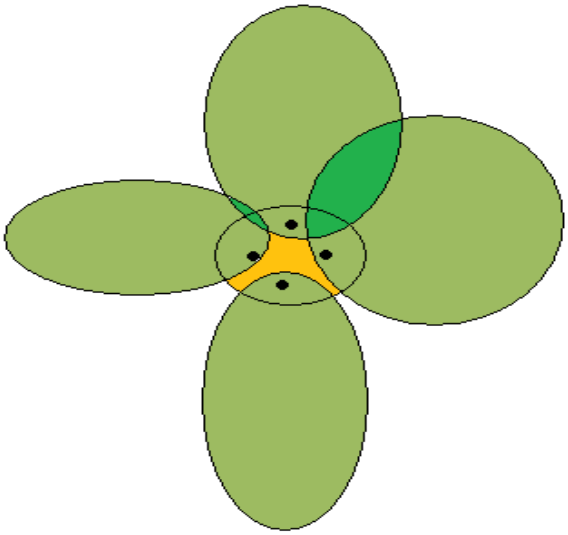

fig 6.Deep data groups representation

- If is under a 33% of hits, it is inferred that the user has not still taken a constant listening attitude maybe because he didn't interacted too much time in something he could find interesting. Then recommendation will try to provide the user with new musical genres he still didn't check.

The selected artist and the selected genre, if present, are stored in the user profile. This information will not be used to perform future recommendations but for loading a user-related home page interface next time he starts a session, as a memory of the previous experience but not affecting the current decisions of the recommender. This way the user is provided with an alternative to start the interaction with the system through these memories instead of searching for a concrete keyword as the very first time. This feature increases general application efficacy by offering the most promising results from the last session as a starting point for the current.

Similar artists' evaluation

This project is not covering how to generate sets of similar artist to a concrete one. By reviewing many papers it was observed that many techniques to fulfill this task have been previously implemented. Some systems, as the reviewed in [12] create a multimedia information retrieval platform, composed by crawlers, which role is to monitor concrete music information web sources like mp3-blogs, Rss feeds and other kind of online documents. Other approach is the selected by Last.fm service. It elaborates complex listening reviews monitoring the interaction of users using their web platform or through the audio-scrobbler[49] plug-in for the local player. Thus they are able to make relations between artist that not always share the same genre specification but still are played by users within a similar context. This interesting grouping of artist has been selected to be used in this project.

The similar artists can be obtained using a function provided by web service API of Last.fm. Play.me service also offers the similar artist's feature but will not be used because the catalog of music available is smaller than Last.fm, thus coming out the problem of musical catalog's intersection. This phenomenon reduces the useful catalog set to the intersection between both services, has been observed to be a surmountable drawback by limiting the artist-album catalog to the very most complete web service.

Redesign of the communication element

Previous prototypes show that Musicbrainz might not be useful, therefore it's wrapper is no longer included inside the communication element. Last.fm web wrapper has been successfully configured, providing good results and complete music catalog. The problem being overcome is the problem of live reproduction of tracks which are not included in the Play.me service, which provides 30 second samples to preview them, but due to the intersection of catalogs problem, this tracks are reduced to the intersection between both services. Thus, this system is providing preview features to a small number of songs, which reduces its utility and avoids meeting initial project aims.

It's not easy to find a web service providing previews for a complete catalog of music for free, but there's a web service that could overcome this situation providing free access to multimedia content. This service is the YouTube video sharing community[50]. It stores millions of videos of songs and other kinds of content, but mainly musical videos, live concerts, and musical related are interesting for this project.

The system's internal logic is conceptually represented in next figure. The Youtube web service acts as a complementary feature loadable when the user decides to interact with a

[49] Project website: www.audioscrobbler.net
[50] www.youtube.com

given track. The video is loaded from Youtube API service and displayed dynamically in web the interface, being possible to be cancelled from the screen in the desired moment.

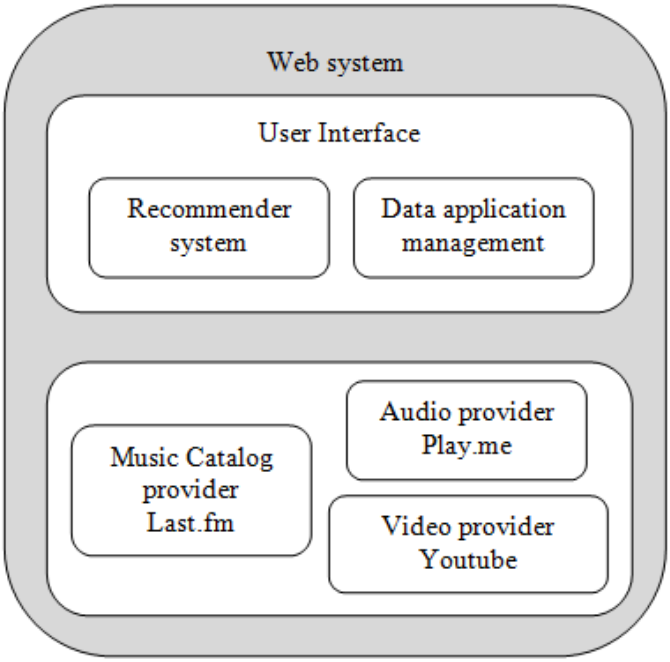

fig 7 schematic figure of web system's structure

Redefinition of user interaction

Changes in recommendation system and in communication element cause some changes in user interaction with in the interface. It has been rejected the possibility of rating an explicit element of the recommendation set. The recommendation is an assessment of implicitly retrieved information, and it has to be evaluated as a whole entity in order to feedback correctly the recommender about the decisions it took. Next table shows how the interaction is understood now.

Action	System response
Input search term	Loads artis and album results regarding the given term.
Artist selection	Loads albums belonging to the given artist and displays some similar artists.
Album selection	Loads tracks belonging to the given album and displays some similar albums.
Track playback	Generates a music player and shows some related tracks
Video playback	Generates a video player and shows some related tracks
Log-in Log out	Loads or stores user profile, and prepares custom interface.

fig 8 User interaction options

Explicit user feedback has been removed from the system, and its information about user preferences is exclusively retrieved from pure implicit actions. How to interpret these actions is the key feature of the recommender algorithm implemented for this model.

Design of user interface

For optimum performance of the application, the system should allow generation of web pages dynamically. This function can be performed using Ajax technology which allows the updating of concrete page components within the client browser, with no need of refreshing the whole page. The website design is done by following a common pattern of content organization. The page is divided into upper interface, for user orientation and arrangement of links to different sections. The bottom part displays the contents obtained each interaction.

The web page obtains the interacted item's *surrogates* and sends them through post requests to the server, where are kept for further review into a user profile-related dynamic structure. This structure is server-side, and its life-time is identical to life-time belonging to the http-session assigned when the user connected to the web platform.

Newt is shown a screenshot of the user interface, showing albums in the lowed side and artists in the upper set.

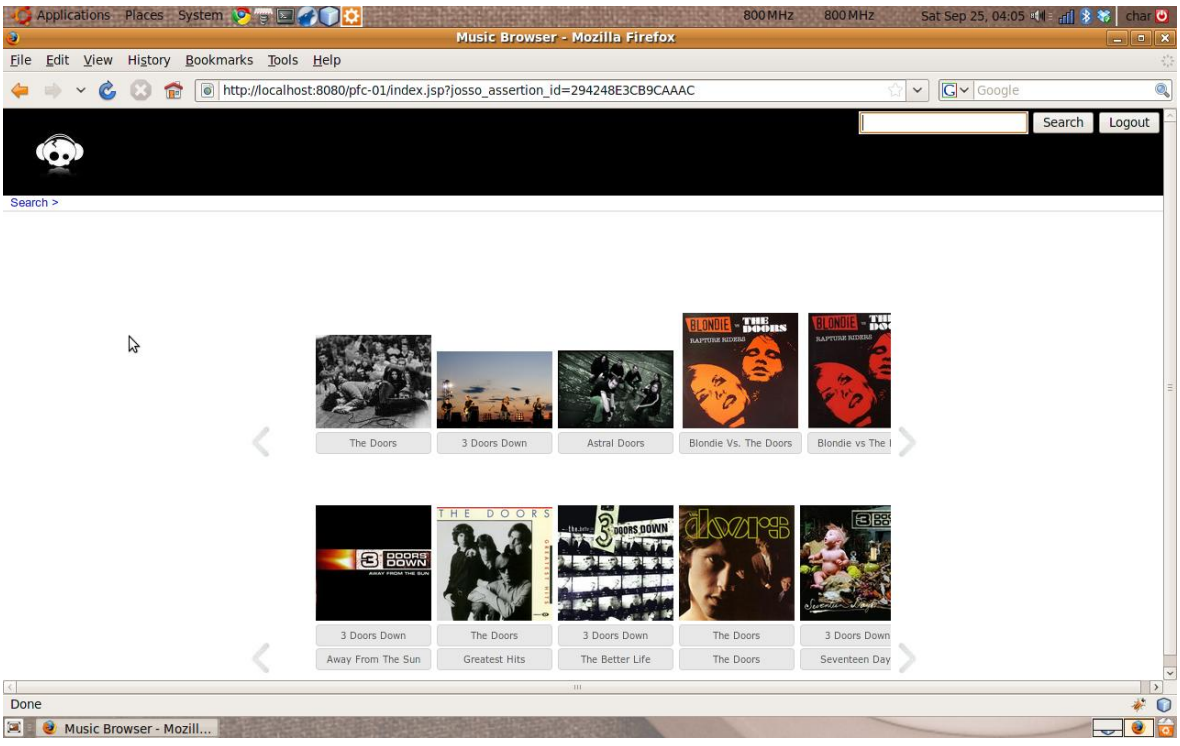

fig 9 Screenshot of the first received results upon search button is clicked

Youtube wrapper prototype

In order to check the Youtube web service[51] features, it has been implemented a wrapper to communicate with is API. This wrapper receives strings of characters as input representing data that should match the retrieved video. The video is the multimedia content from Youtube[52] used to satisfy a track preview. Keywords are searched along the video attributes, inside the video name, or inside some of its possible tags, which are not always present. The keywords are mainly matched with the words listed in the video name. This matching seems to be consistent during testing. But it is acknowledged that it's possible to provide some track name, which includes some detail about album release on it, making difficult deal to match some video for a complex name. For example:

For the track name: The doors – Alabama song
Is it possible to find in some releases: The doors – Alabama Song (deluxe remastered)

It is convenient to solve this problem by parsing previously the track name sent to the wrapper, with special focus in prevail the chance of getting a response while forgetting a little bit about strict video matching.

Web system prototype

The web system is characterized as a multiuser system with access to shared resources, data persistence system, and dynamic generation of the content displayed to the user. It has a dynamic interface that is generated based on customer interaction with the application. User actions are transferred to the web system using Ajax. The web system receives post requests from the web browser and redirects them as http-requests to the web services thought the wrapper implementations. The responses coming from the web services are analyzed by the web system, storing some information from them in server-sided structures for performance purposes and sending them back to the client encapsulated in Json objects. Json was previously reviewed as a powerful and simple object notation.

The application is hosted in a Tomcat server[53]. It aims to manage multiple sessions of different users connected at the same time. Therefore concurrency strategies are developed. This ensures consistency of data stored in the servlet shared memory. The structure of Java servlet provides a simple way to implement web support with the greatest flexibility available, since the servlet is the simplest web entity out there. There exist other web programming languages which offer better solutions for encoding and simplicity, such as Php[54] or Python[55]. But Java[56], despite its awful coding, has a large number of modules and

[51] code.google.com/apis/youtube/overview.html
[52] www.youtube.com
[53] tomcat.apache.org
[54] www.php.net

functions already implemented with reliable code and compatibility issues solved which are quite time-saving advantages.

The system designed handles multiple servlet. The servlet, such as simple web unit, does not implement user management (shared memory, concurrent access), but accepts connections from different clients running concurrently. The method in which the call is executed from the client browser http, poses serious concurrency problems when using shared variables.

To overcome shared-memory problem, we have implemented a simple method for controlling access to shared memory to ensure consistency of the data used by the application. The access is synchronized so that the loading and writing of user memory is done under mutual exclusion. Access to customer-specific memory is indexed through the session identifier assigned to each customer connected, ensuring access to a particular area of memory for each thread running within the servlet shared method.

The web system has three servlets which are assigned different tasks.

One of them **manages user's connections** with the web service related to music catalog retrieval. The catalog is used to provide music information to the user, like artists, albums or tracks. This information is obtained from the music service Last.fm.

Another servlet is responsible of **retrieving musical collections** for the recommendation system. System actions that are not explicit user commands are executed by this servlet. Requests are sent to the web service of Last.fm. In order to improve overall system performance, affected by excessive competition over too much synchronized blocks of code, to share the tasks in two servlets lets shorter response times.

The third servlet **handles communication with the API of the video** service provider, allowing the catalog collection requests and video results requests are made jointly, providing clear benefits in system performance. Instead the music streams from Play.me web service can be requested using the API created for this purpose. It directly handles post requests building up the correspondent call to this web service. It has been included a complete control of exceptions which occur upon incomplete responses, mainly launched by the Json parser. This API provides robust and safe working due to its concretely defined purpose.

In this step, has been developed the entire web system. It has been added the recommender functionality plugging the functions within the server-side servlet files. This extension gives rise to the User class, which includes the features necessary for the recommendation feature, user-data management, user-related recommendation engine, and temporary server-side user-data.

[55] www.python.org
[56] java.sun.com

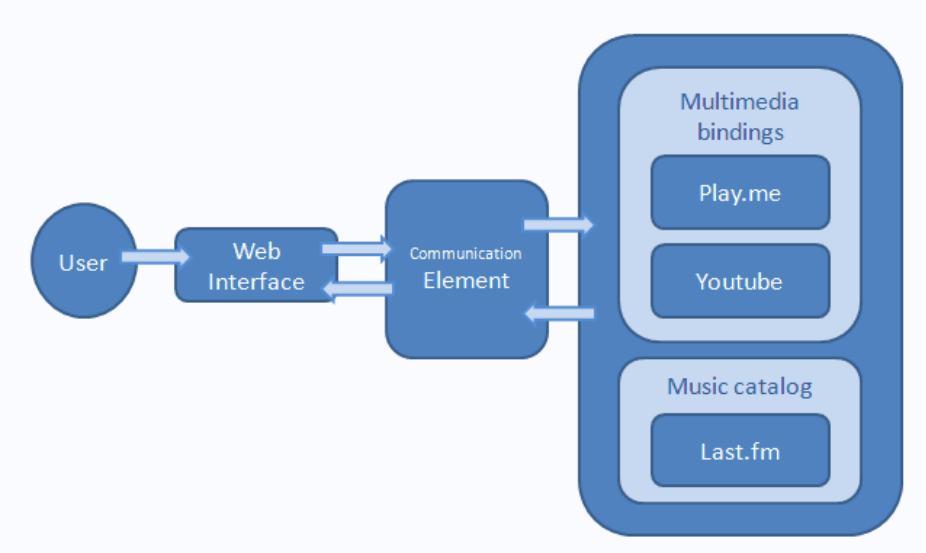

fig 10 Web system's prototype implemented for this iteration

Conclusions

Testing of the whole system has been performed obtaining satisfactory results regarding the running of implemented wrappers. The music catalog is loaded normally, and no latency problems have been observed using a discrete internet connection of about 2Mb of bandwidth.

It is possible to interact with the system in a loop that lets the user navigate continuously with no need of inserting any text, only through mouse clicks. This feature clearly enhances the application usability greatly easing the user interaction, while providing rich multimedia information.

Some problems have been found regarding the recommendation event, and so regarding the way in which the interface is organized. These problems are not really related to system design but yes to some aspects which could not be detected upon test with real users. Suggestions from users which interacted with the system in this beta version assessed some interface problems. For example the recommendation set comes out in the lower side on the window, as shown in the next figure. This fact requires the user to scroll down the web page in order to check the recommendation content, which is announced by a pop-up kindly asking the user to check the new musical stuff.

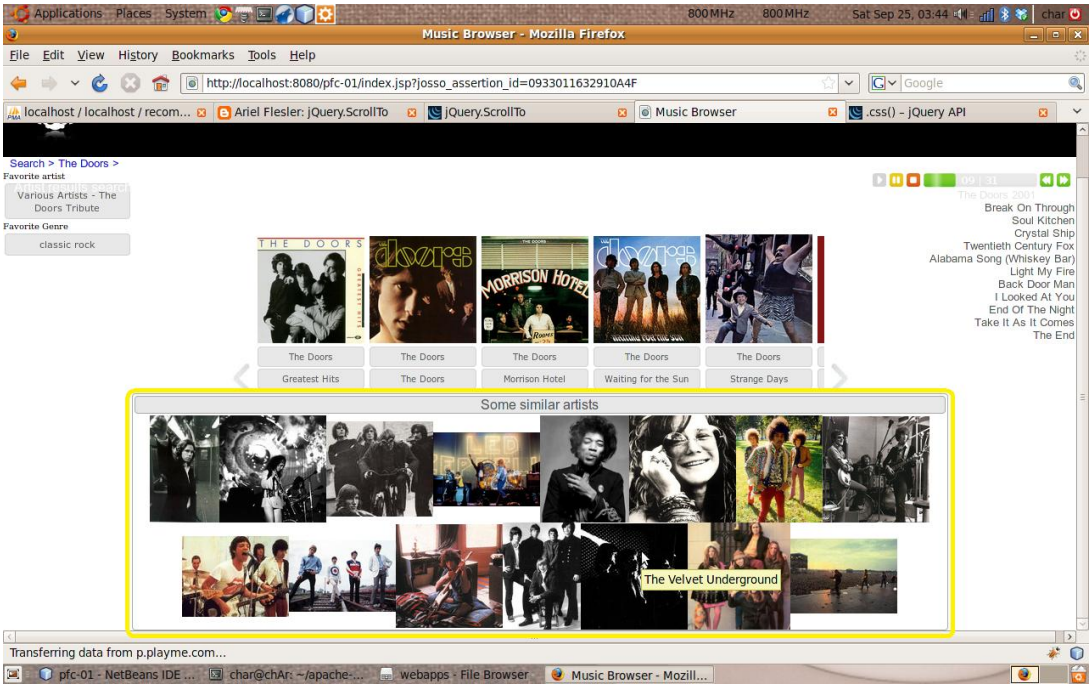

fig 11Interface event, recommended content highlighted in yellow color.

The recommender engine is tracked, observing that sometimes, albums or artist do not include genre definition, due to lack of tags in the music service. In earlier stages of developing occurred a similar issue relating the release date of albums. Most of them were not including this data even if in the data-object specification was clearly mentioned to be present. This event led me to forget about playing with dates and époques for creating recommendations, due to its lack of reliability. Current problem with genre is not as severe because it only affects a small number of groups which are not very popular or have been recently added to their musical database.

Recommendations are built on-the-flow with data collected on-the-flow, and no further storage of information is persisted with recommendation purposes. In fact recommendations suit the user likes but still do not satisfy the project aim and further enhance tends to be used.

6.4. Third Iteration

Objectives

The objective for this iteration is mainly to enhance the whole system, which is observed to work properly but greatly improvable still to be left as it is as final system. In order to achieve better recommendations, an alternative to the proposed algorithm would be found. Furthermore, extensions in the persistence system can be done in order to store part of the information the system needs o create recommendations. This extension will be studied along this iteration assessing its advantages and disadvantages.

A better user interface is to be designed, once the problems related previous design have been identified, won't be difficult to achieve improvements in this matter.

Control all the bugs and exceptions emerging to the web interface will be other objective covered in this iteration. Some events which fire upon abnormal web services responses are not easy to handle, but enhancing the exception control, no error pop-ups or blank results page would be displayed.

Algorithm analysis

Most of the reviewed systems rely on collaborative filtering to solve problems like the unrated item problem described in [7]. The previous algorithm could be defined as a content-based like with some influence of context-based recommenders, overcomes this problem by including non previously interacted (rated in other approaches) genres or artists. It is done when no relevant information is found inside the current interactions data-set. The approach is to select music genres which are not present within this data-set and elaborate the recommendation based on these genres.

Focusing the problem

The problem still pending to be solved relies on the fact that the retrieved genres could be completely disliked by the user. It will be called the problem of *blind* recommendation. This might happen as a direct consequence of the recommendation method itself. The method completely forgets about user's preferences stored in the profile and looks for new user genres using the web services. It may be a significant improvement to include in this case, some details regarding user preferences, which might be inferred using different techniques or extending the present approach.

Solving alternatives

Many different alternatives to implement the recommender procedure can be found in [14] often based on combinations of collaborative-like and content-based-like approaches. It

seems interesting for this recommendation issue the fact of scoring each user selection, for example, by giving higher ranks to items clicked more recently while lowering the rankings for the most distant in terms of time. Including this ranking system, it is possible to reload previously high-clicked items when no relevant information found inside the monitored interaction. Thus, the *blind* recommendation can now be based in some item, that maybe does not seem interesting to the user currently, but at least it can infer that in some moment it was, avoiding to provide completely uncorrelated items according user preferences.

Analysis of user management solutions

There are several ways to solve the user management in a web site. There exist a variety of modules that can be plugged, Java libraries, etc. The system needs an element that checks the user's identity, preventing access to application data to any user without valid credentials for access. The element must be incorporated into the system without making major modifications to either element.

Several modular solutions have been studied, assessing the complexity of installation, ease to configure, the type of server they use and the problems of incompatibility the database system version used. Among them the most outstanding are Josso, JFacets and OSUser. All of them are "easy" to install and configure. Josso[57] was chosen because of its apparent simplicity, it is providing various security protocols, and as main feature +, the functions it provides match the needs of this project and are written in the same J2EE programming language used in the web system.

Josso (Java Open Single Sign-On) is a smart gateway to one or multiple web applications. Josso handles all the connections from one or more log-in web pages which give access to some application controlled with it. In these log-in pages, the user enters the log-in details, which are sent directly to Josso. Josso checks the user credentials and sets the data source connection. Once identified the user, Josso publishes an entrance ticket which is valid only for the user credentials and for the http-session that requested the ticket. The ticket allows access to the system protected areas (declared in xml files) , these areas are just folders containing the code that allows the execution of the music recommendation system.

[57] www.josso.org

Design of new algorithm

For the algorithm design previous problems tend to be determined precisely. Then some solutions for each determined problem will be provided in order to enhance significantly the previous lacks in recommendation precision.

As a previous problem it has been identified that sometimes the implemented algorithm completely forgets about user preferences as a try to provide him with undiscovered music genres. This behavior is not counter-productive, because many genres will be provided being very improbable every genre is completely disliked by the user. But it still breaks with the finer recommendation politics and maybe cross a conceptual border which should be strictly respected.

A limitation of the previous implemented model relies on the impossibility of performing searches by genre for example among the results previously loaded. It is possible to request to Last.fm an artist collection given a genre , but this results, are ranked results, and the method does not retrieve all the available elements, just the top tagged using that genre tag . It is possible to store some other information about user interaction inside the database system. Each time a user interacts with some music element , classified using the tag this interaction will be stored in the database. As more times the same interaction occurs more important will be that element for that user. It can be also stored the element surrogate genre in order to classify information for further use of the recommender system. Upon each user interaction, an interacted element will be associated to its tag and associated to this user. This association can be stored as our own genre classification for the interacted artists.

Then would be easier for the recommender to collect information about user's likes even to start a collaborative-like environment in which it is known which user has interacted with which artist and when. Further information can be stored like playback environment in order to specify more clearly the context in which the interaction occurs. By including this new feature, the use of some memory about the evolution of the user preferences is possible. Using the a artist-genre-user-time relationship table, is feasible to keep a long-term memory about user interaction as the time the previous monitoring structure caring about mid-term memory is still used. Now is it possible to give preference to the most recent user events ( ), as project's aim remarks, the use of older events to extract information will happen if the mid-term memory is insufficient or is corrupted (lack of some attribute).

Next is presented the algorithm applied for this third iteration prototype. Now we can speak about the type of user-interaction information stored, long-term memory which is stored in the database table where relations between information objects are stored. Mid-term memory represented by the selections taken by the user which are buffered in the temporal user application data.

Assumptions

A hit is a click over a musical surrogate[58].

The long-term memory is stored permanently LTM. Stores relations between artists and interacted by all the system's users.

The mid-term memory or MTM is stored temporally until the http session expires.

Each click refers to a musical surrogate independent from the item (artist, album or track) that generated it. The surrogate includes artist name and musical genre. Due to reasons beyond this system's capabilities, the year or époque related to the musical item could not be taken into account for recommendation purposes because it is absent in most of the musical elements checked.

Mid-term memory analysis

Once the configured threshold is reached, the recommendation is activated. A statistic procedure is applied over the collected data in similar way as before.

First it's checked whether exists some artist mainly repeated (most than the 50% of hits) In the case it's found, similar artists are retrieved from the catalog provider. In the other hand, a query to the database is performed to obtain the system's registered information about the given artist. From this information the genres associated to the given artist are used to retrieve all the artists which are referenced thought some of those musical genres.

In this procedure, two data-sets of artists can be managed. It provides many possibilities for using this information for recommendation purposes. For this algorithm, an intersection between both data-sets is performed. Then a minimum and a maximum threshold of retrieved results is set.

If the intersection does not meet the minimum threshold then further decisions can be taken, next step is described in the *upon genre selection* section. At this point, the algorithm conceptually diverges from the previous proposal. Now the musical genre is always studied to perform recommendation while in the previous was only utilized when no favorite artist was found.

The procedure to get the previous most repeated genre is performed almost as before, excepting the new concern about the time line of actions. There is a direct inverse relationship between the importance of a given selection's score (number of hits in time) and the time in which they were registered. The actual procedure to get the main genre is explained in the *second iteration, first algorithm proposal* section.

[58] It has been defined surrogate as the inferred attributes from a musical element.

LTM memory analysis

In the case the genre searched in the MTM doesn't arrive to the 33% of hits, the behavior changes and the LTM memory is analyzed. It's retrieved the maximum rated genre as follows:

Parameter can be configured to assess the importance of the recent events over past events.

This method cannot success in the case no LTM is available (new user case) In this case, the recommendation will be performed as described in the second iteration. In case of success (the rest of cases) the selected genre is put together with the similar genres retrieved from Last.fm web service.

Upon genre selection

The number of genres is also determined by the parameter, as also is the number of further artists requested for each musical genre by the parameter. Once a main genre is found by some of the appropriate procedures, and in the case that the main artist has been successfully found, both data-sets of artists, one from the LTM register , and other , from the Last.fm web services are filtered.

All the artists belonging LTM data-set classified with the musical genre are selected for recommendation. Artists tagged with are retrieved from Last.fm. The results received are compared with the previous collection containing main artist's similar set . The elements remaining in the intersection of these two collections are added to the previous collection in order to complete the recommendation reaching the maximum threshold for the response dataset.

Conclusions

This algorithm ensures a optimal recommendation based on metadata information of music, according the preferences inferred from the monitored user interaction. New interesting music ins provided to the user. The tests have been performed in sessions of maximum fifteen minutes of interaction and the recommendations received provided the user with new music which *surrogates* have never been classified before. This is an important feature about discovering which has been fulfilled.

The model used to overcome this problem has been the algebraic groups theory[59], which optimally represents the abstraction of music collections retrieved from music services.

[59] www.jmilne.org/math/CourseNotes/GT.pdf

Redesign of user interface

The user interface has been redefined to solve the problems meet the previous interaction. Thanks to the testing elaborated by people extern to this project, some improvements can be assessed for this new version of the interface.

This new version tends to organize all the necessary information in such a way the user is not forced to scroll up and down the interface to check, at least the recommended items. It's also needed to include some extra information about the displayed information. It seems that a new user could not easily understand what is happening in the interface when he clicks in one item, but conveniently explaining the content this overcome will be solved.

The new web interface is created using design-oriented software, designing each component separately and them putting them together inside the html document.

Plugging the Josso module

Josso module configuration is simple in appearance. It is made up by an agent, and a gateway. These two elements are configured using xml files to be which contain information concerning the system settings.

The agent is responsible, among other things, of the access configuration to services related to persistence. The Josso-related stored data, refers to access credentials for each registered user and web system-related characteristics which must be remembered between sessions.

The gateway defines the type of security employed, the address relative to the server assigned to redirect user connections after accessing the system, also defines redirections to error pages, or redirections to the log-out page.

Another important aspect of the Josso module connection is the behavior for new accounts, or password recovery. Josso does not allow access to protected areas upon the user access the system successfully through the unique ticket generated by Josso. Until then, Josso protected content remains locked. In order to store the log-in page which is accessed freely but is still store inside the project that Josso controls, it's needed to create a public folder. This folder can be assigned reading rights not using the Josso unique ticket. In this folder will also be stored pages concerning registration or password recovering as well as further dependencies like javascript files or css rulers which are needed to correctly display public pages.

To enable routing to several pages, such as error pages or password recovery, has been designed a new servlet. Its task is to handle requests related to registration, to retrieve new account data or password. It provides access to the login page when user registration data meets the Josso constraints.

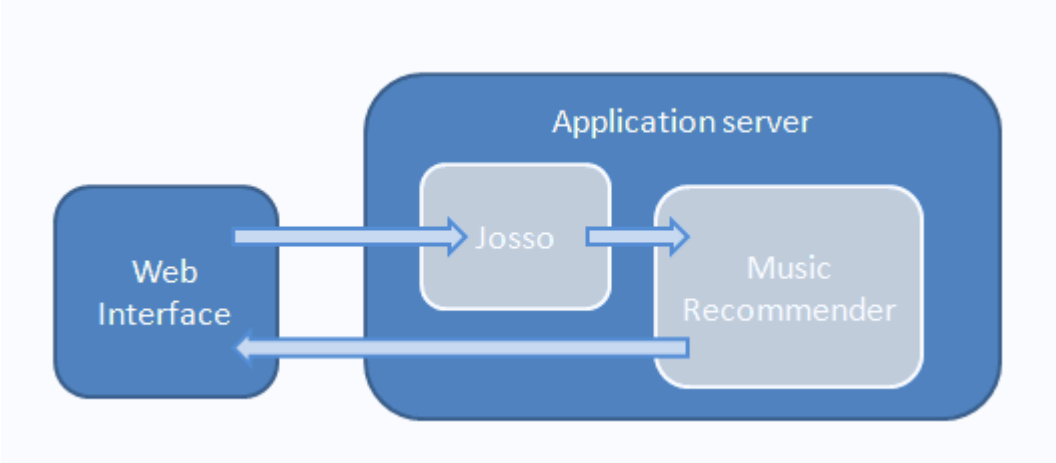

fig 12 Josso module placement inside the system

Expanded prototype implementing user management

The final system implemented for testing is installed on the server, together with the Josso module to provide users management. Josso ticket has a unique identification which is used to store the user-related temporal data. A pool of connected users is stored in the server. This pool structure is a *key*, *value* pair-like structure storing the application data for each user, indexed using the Josso unique id.

The user-related application data has the goal to complete the musical elements received by a search for artist or album. These are returned incomplete for music services and must be completed to meet the application requisites. As explained in the section of prototypes belonging to the first stage of development, to complete the music object with which the program works, it is required a second request to the information provider. This request can be made asynchronously to avoid latency, thus allowing the user to continue the browsing while the system processes the other requests.

Other important feature of user application data is to save some requests to the music providers when previously loaded items are requested again. This data is stored temporally for each user as a measure that enhances system's performance.

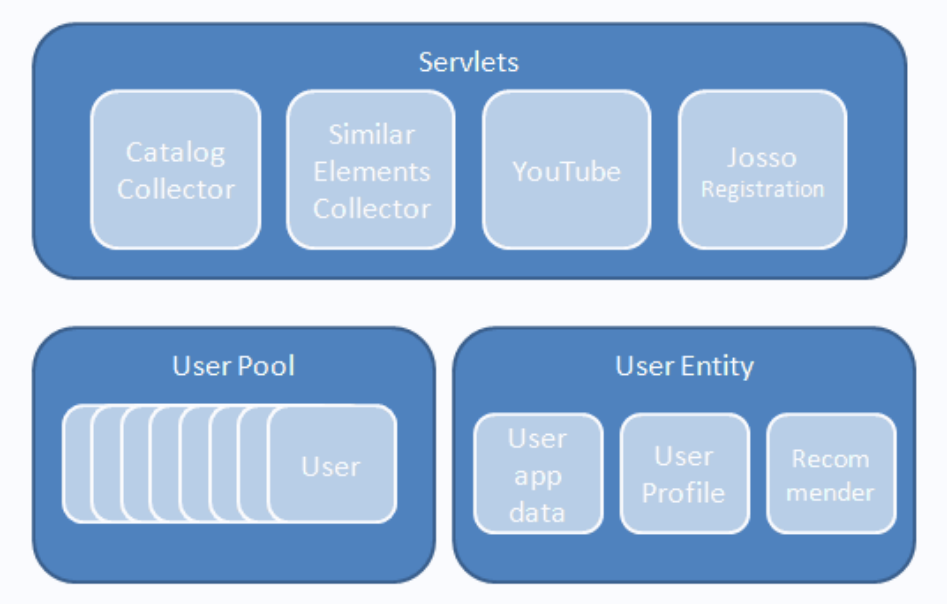

fig 13 We system overview after adding user related content structure

The user interface is implemented using JSP pages that provide the functionality to run code on the server, useful for extracting information about the session started by Josso and its attributes. In this way the user is easily be identified in the system without other java servlets that cover this actions.

The dynamic activity characterizing the interface is achieved using libraries written in Javascript language[60]. In particular, Jquery[61] supports the Ajax functionality. It's a library that provides a useful framework simple and powerful. It stands as a bridge between the web browser post requests and the server logic (servlets). Also cares about reception and reconstruction of some elements encoded in json returned by the web system. Web browser-music services communication is completely developed using this library. Jquery is also used to generate the visual effects of the interface and bridges dynamically generated DOM elements and their css style definition.

Data persistence tests

There has been created a custom web interface which served as a testing environment for the persistence system. The selected open source technology is Mysql[62], mainly chosen due to its easy to use interface called phpmyadmin[63] which is a complete tool to manage the database system and generate complex relational databases in a reliable way.

Upon this tests were passed the methods utilized were plugged inside the code, providing persistence features for data-types used by this application.

[60] Javascript definition: en.wikipedia.org/wiki/JavaScript
[61] jquery.com
[62] www.mysql.com
[63] www.phpmyadmin.net

7. Conclusions

Project objectives

The system development has been completed successfully. Its usefulness for discovering new music has been tested meeting and the goals stated at project's objectives.

It has been proven that is not needed to build a huge information system to provide the user with new music that matches his likes. Taking advantage of available web services which provide complete music catalogs for non-commercial purposes.

However, it was observed that the development of a recommendation system with commercial features actually requires an extensive relational database to store the music previously cataloged. Creating a good relational database of music, making relations between artists, albums, musical genres and époques, could greatly expand the capabilities of a recommender algorithm to help users discover new music.

Purpose remarks

An important point to comment is that while all commercial recommenders claim to be offering *personalized* recommendations, the truly nature of their predictions might be biased by the taste of the majority of users, instead being biased by the actual user's preferences.

This has been an important assessment made at the project work-lines definition, to focus mainly in authentic user preferences, and furthermore making especial care about the earliest implicit tips extracted from user actions. The real motivation which pushed me to consider this approach should be mentioned:

- On one hand, my personal view before and after this thesis work about music recommenders is that they should meet user's preferences whatever approach is implemented to do so. Relying in *collaborative filtering* techniques is a risky issue because it slightly diverges from pure user interests, by abstracting the user preference pattern to meet some similar users. This abstraction is the actual loose of information which generates the initial purpose deviation.
- On the other hand, the *collaborative filtering* is a way of abstracting some king of *music hierarchical order* in which users implicitly decide what's the best music while their actions are monitored. This fact is strongly determined by the music industry and its merchandising policies. Their promoting activities of some *economically remarkable* groups or artists lead people to subliminally remember them, thus seeding in customer's minds a spot of interest related to this groups or artists.

Technical remarks

Popularize the use of the fingerprint is one of the most promising options for the future of music recommendation. The possibility of combining the techniques of collaborative recommendation, with data-mining techniques applied on the data collected from user monitoring, is an interesting path to infer optimally user's preferences. Subsequently, adding the ability to recommend music similar (or identical or different) according acoustic features, means that future recommenders could provide us with the music we really want.

Utopical facts

The ultimate recommender could provide the user with a different rock song (for assessing a concrete example), featuring a slightly faster tempo (compared to the previous song) but with more folklorically-liked instrumentation. Such query might be a common music query for future recommenders.

8. Contributions

Reviews

A complete 2. The recommender systemsabout the state-of-the-art of recommender systems, specially focusing on music recommender systems has been performed. Aimed to settle the project analysis and design strategies, as well as define correctly the work outlines.

In order to provide the user with a free, complete musical catalog, a revision of music web services has been performed. Through this work it has been assessed which service provides better suited solutions, starting a deeper analysis of them by implementing prototype wrappers to enable communication with this web system. As a result of this deeper analysis, the web system successfully achieves the communication functionality by providing music data from the selected web services for its use within the implemented environment.

Code

Code reusability is a key decision factor regarding the framework selection. So, the project will be implemented using Java programming language. This time-saving advantage benefited the project when some BSD licensed-software components were found to be developed in the selected language. Some custom adaptations or code modifications were needed in order to successfully plug these extern modules inside the project definition.

Unfortunately most of the code pieces needed to build up this work are not previously implemented. A new wrapper has been implemented supporting some required methods employed to retrieve musical data from Play.me web service.

Other implemented wrapper allows communication with the video service of Youtube. Its main feature is the track name parsing, oriented to properly select the best suited video from a feed, composed by thousands of possible matches.

Recommender algorithms

Upon a data domain was defined, a first custom designed algorithm was implemented from scratch. This algorithm (6.3. Second Iteration was designated to cover well defined recommendation constraints using custom defined item *surrogates*. Its recommender helpfulness was probed using empiric tests using random users, but were also assessed some limitations and deviations from the very main recommendation purpose. A further study, based on the current data-object *surrogates* and system characteristics, gave rise to a new evolved version of recommendation algorithm. The second algorithm proposal provides solution to problems observed theoretically after successful empirical tests over the 4. First Algorithm proposal

The second algorithm was designed using algebraic grouping theory for solving some

use-cases where the recommendation was ignoring few possible positive responses. Reviewed in previous works, the recommender problems related to selected approaches (*see collaborative and content based filtering in section 3.2.* Approaches to music recommendation) were indentified. This second version has been designed to solve these problems in an effort to develop a fine recommender. Grouping theory suits well this recommendation approach: The collection oriented structure of Last.fm responses, can be abstracted as a group, whose membership constraints are based on musical features of objects.

The whole approach

The system overview itself is an innovative approach to music browsing and discovering including efficient recommendation features. The system is completely implemented using open source tools and modules. It is convenient to remark that the system exploits some resources freely provided by commercial music systems required to achieve its functionality.

9. References

[1] Montaner, M.; Lopez, B.; de la Rosa, J. L. (June 2003) "A Taxonomy of Recommender Agents on the Internet", *Artificial Intelligence Review* **19** (4): 285–330, http://eia.udg.es/~blopez/pub.html Sat, 23-10-2010

[2] http://labs.oracle.com/projects/dashboard.php?id=153 Sat, 23-10-2010. Search Inside the Music is a project of Sun Labs, Burlington, MA.

[3] http://www.moyak.com/papers/collaborative-filtering.html Sat, 23-10-2010

[4] Using Content-Based Filtering for Recommendation. Robin van Meteren, NetlinQ Group, Amsterdam, and Maarten van Someren, University of Amsterdam, The Netherlands.

[5] http://www.pocket-lint.com/news/29588/itunes-most-popular-music-service Sat, 23-10-2010

[6] Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions, Gediminas Adomavicius, Alexander Tuzhilin, IEEE Members

[7] Creating a Hybrid Music Recommendation System from Content and Social-Based Algorithms Jamie Cai, John Francis, Stephen Gheysens, Rutgers University, GSET '09

[8] http://en.wikipedia.org/wiki/Acoustic_fingerprint, Sun, 24-10-2010

[9] http://marsyas.info, Sun 24-10-2010

[10] http://en.wikipedia.org/wiki/Timbre, Tue, 26-10-2010

[11] Belkin, N. J., and Croft, W. B. Information filtering and information retrieval: Two sides of the same coin? Communications of the ACM 35, 12 (December 1992), 29–39.

[12] Foafing the music: music recommendation system based on rss feeds and user preferences.

[13] Influence in Ratings-Based Recommender Systems: An Algorithm-Independent Approach Al Mamunur Rashid, George Karypis.

[14] Shardanand, U., and Maes, P. Social information filtering: Algorithms for automating "word of mouth". In Proceedings of CHI'95 (1995).

[15] Jensen, F. V. Bayesian Networks and Decision Graphs. Springer-Verlag, 2001.

[16] A music recommendation system based on music data grouping and user interests Hung-Chen Chen and Arbee L.P. Chen Department of Computer Science National Tsing Hua University Hsinchu, Taiwan 300, R.O.C.

[17] Classification of recommender systems according to knowledge representation criteria, Content Based Recommender, M. Ramirez, June 2005

[18] Exact algorithms for NP-Hard problems: A survey, Gerard J.Woeginger, University of Twente, The Netherlands.

[19] User modeling via stereotypes. Cognitive Science, Rich, E.(1979), 329–354.

[20] Powell, M. Approximation Theory and Methods. Cambridge University Press, 1981.

[21] Murthi, B. P. S., and S. Sarkar, The role of the management sciences in research on personalization. Management Science 49, 10 (2003), 1344–1362.

[22] PSUN: A Profiling System for Usenet News (Extended Abstract) H. Sorensen, M. Mc Elligott, Computer Science Department, University College, Cork, Ireland.

[23] Music Recommendation by Modeling User's Preferred Perspectives of Content, Singer/Genre and Popularity Zehra Cataltepe and Berna Altinel Istanbul Technical University Computer, Istanbul, Turkey (Review section)

[24] Probabilistic Models for Unified Collaborative and Content-Based Recommendation in Sparse-Data Environments, Department of Computer & Information Science.

[25] Collaborative Filtering for Information Recommendation Systems Anne Yun-An Chen and Dennis McLeod, University of Southern California, Los Angeles, California, USA

[26] An algorithm for finding nearest neighbors in constant average time, E. Vidal Ruiz, Universidad de Valencia, Spain, 1995.

[27] Najarian, K. and Darvish, A. 2006. Maximum Likelihood Estimation. Wiley Encyclopedia of Biomedical Engineering.

[28] Content-based recommendation systems, Michael J. Pazzani, Rutgers University, ASBIII, New Brunswick, NJ Daniel Billsus FX Palo Alto Laboratory, Inc., Palo Alto, CA

10. Referenced links

1. Amazon website: www.amazon.com
2. Jamendo, Open Source music recommender: www.jamendo.com
3. Last.fm music recommender: www.last.fm
4. iRATE is a open source music recommender: irate.sourceforge.net
5. MLEstimation review: mercury.bio.uaf.edu/courses/wlf625/readings/MLEstimation.pdf
6. Hard Np-problem clearing: www.math.ohiou.edu/~just/bioinfo05/supplements/Lect_NP.ppt
7. Pandora recommender system: website: www.pandora.com
8. Spotify music streaming desktop system: www.spotify.com
9. Magnatune radio website: magnatune.com
10. Apple Itunes Store: www.apple.com/itunes/what-is/store.html
11. Music Recomender: www.emergentmusic.com
12. Audioscrobbler free opensource collaborative engine: www.audioscrobbler.net
13. Api documentation Jamendo web service: developer.jamendo.com/en/wiki/MusiclistApi_draft
14. FOAFing the music Project: foafing-the-music.iua.upf.edu
15. Information retriever Pubsub Website: www.pubsub.com
16. Emusic music web service: www.emusic.com
17. Yahoo music web service: new.music.yahoo.com
18. Soundcloud music web service: soundcloud.com
19. Discogs music web service: www.discogs.com
20. Rhapsody web service: www.rhapsody.com
21. Musicbrainz project web site: musicbrainz.org
22. Play.me music web service: www.playme.com
23. Content management system: www.magnolia-cms.com
24. Content management system: atleap.dev.java.net
25. Content management system: www.pligg.com
26. Last.fm api documentation: www.lastfm.es/api/intro
27. Review about CreativeCommons licenses: creativecommons.org/licenses/BSD
28. Last.fm java bindings: code.google.com/p/lastfm-java
29. Servlet technology review: www.oracle.com/technetwork/java/index-jsp-135475.html
30. Ajax technology web site: www.ajax.org
31. Javat serializable object notation: www.json.org

32. Musicbrainz xml web service: musicbrainz.org/doc/XMLWebService
33. Musicbrainz java code: sourceforge.net/projects/javamusicbrainz
34. Play.me web service API: lab.playme.com/api_overview
35. Audio fingerprinting inside Musicbrainz: wiki.musicbrainz.org/AudioFingerprint
36. knn-algorithm explanation: www.lkozma.net/knn2.pdf
37. Bayesian networks explanation: www.eng.tau.ac.il/~bengal/BN.pdf
38. Joint Probability Distribution: www.stat.tamu.edu/~henrik/211S05/notes/chp5.pdf
39. Laplace principle of inference: en.wikipedia.org/wiki/Principle_of_indifference
40. Youtube video web service: www.youtube.com
41. Youtube apio bindings: code.google.com/apis/youtube/overview.html
42. Apache server technology: tomcat.apache.org
43. Php scripting language: www.php.net
44. Python scripting language: www.python.org
45. Java technologies web site: java.sun.com
46. Java open-single sign-on security module: www.josso.org
47. Grouping theory overview: www.jmilne.org/math/CourseNotes/GT.pdf
48. Javascript language definition: en.wikipedia.org/wiki/JavaScript
49. Jquery web framework: jquery.com
50. Mysql database systems technology: www.mysql.com
51. Mysql database manager: www.phpmyadmin.net

11. List of figures