Context-Based Collaborative Recommendation System to Recommend Music

Zapata C. Santiago Ph.D., Escobar R. Luis, Águila G. Elías
Department of Computer Science, Engineering Faculty, Metropolitan Technological University
Santiago, Chile
szapata@utem.cl, laescoba@utem.cl, elias.aguila@gmail.com, www.utem.cl

Abstract - A recommender system based on user collaboration partners in communities of a social network, can build a collective knowledge to help recommend automatically lists content to users of a social platform, based on their behavior and preferences. The purpose of this paper is centered in implementing an Android Recommender System capable of providing user's songs without being assessed by the user, are estimated to be on their taste. To do this, the application is made around a Collaborative Recommendation System, following the entire system as a whole client/server architecture.

Keywords: Fuzzy Logic, Data Mining, Fuzzy Controller, Fuzzy Rules

1. Introduction

In today's time, information overload is becoming increasingly apparent as more and more data sources are accessed. On a daily basis, users are faced with the challenge of choosing an application for their Smartphone or choosing a good movie to see the weekend between thousands and thousands of options. This is the main motivation scenario of the "Recommendation Systems", a set of tools that seek to reduce the user's cognitive effort by studying patterns of behavior that allow predicting the possible choices that a person would make among a set of items with which you do not have previous experience.

2. Research Topic

Consider a Recommendation System to recommend music, which is precisely our case study: a user with certain preferences could receive different recommendations depending on their status, whether they are working, resting, performing physical exercise, etc. These are aspects of the user context, which can be taken into account by a recommendation system. Context-based Recommendation Systems are systems that address these variable circumstances that pertain to the user's spatial and temporal environment, geographic location, day of the week, season, etc., and in particular it could be considered as belonging to the context the type of activity developed: working, resting, exercising, etc. In this research we will focus on the problem of recommending music following a hybrid collaborative approach with a context-based approach, which mainly contemplates the type of activity that the user is performing: running, walking, sleeping, working, studying, buying, etc.

3. Theoretical Framework

Systems Recommendation (SR) are techniques and software tools that provide suggestions of items that are of use to a user. The suggestions relate to several decision processes, such as what items to buy, what music to listen to or what news to read online. [2] "Item" is the term used to denote what the system recommends to users. An SR typically focuses on a specific type of item (such as CDs or news) and according to its design, GUI and recommendation technique used to generate recommendations are customized to provide effective and useful suggestions for that specific type of item.

SRs are primarily aimed at individuals who lack sufficient personal experience or competence to evaluate the potentially overwhelming number of alternative items that a website, for example, can offer. A specific case is a book recommendation system that assists users in selecting a book to read. On the popular Amazon.com website, an SR is used to personalize the online store for each customer. As they are personalized recommendations, different users receive different suggestions.

In addition, there are also non-personalized recommendations. There are simpler suggestions to generate and are usually named in magazines or newspapers. Typical examples include the top ten selections of books, artists, etc. While they can be useful and effective in certain situations, these types of non-personalized suggestions are not commonly the target of investigations of recommendation systems.
3.1 Basic Scheme of a Recommendation System

The basic scheme of an SR exists the following main elements:

- **Database**: The quality of the data stored in our database plays a fundamental role in making recommendations with higher or lower quality.
- **User Profiles**: A user “shapes” your personal profile as you use the system. The profile reflects the tastes/preferences of the user, fundamental when discriminating objects during the recommendation.
- **Predictions**: Prediction plays a crucial role within the basic schema of every SR. The prediction is based on the profile of the user and the information available in the database that is taken into account.

3.2 Data in the Recommendation Systems

It is important to make some decisions when developing an SR, such as the type of feedback used, the type of data to use or how these data will be analyzed.

**Feedback on Recommendation Systems**

An SR should not be a static entity, but the effectiveness of its suggestions must evolve with the passage of time, based on the experience and new information obtained. This is achieved by applying feedback mechanisms between the system and user preferences. For this, there are two mechanisms of feedback: implicit feedback and explicit feedback.

**Realimentación Implicita**

An implicit feedback mechanism is one that provides the SR with information about user preferences without being aware of it. These feedbacks are not done directly, but through some measures such as: the time of visualization of the object, the number of queries, etc. It presents the problem that it depends too much on the context and is quite hypothetical, since assumptions are made (based on the mentioned measures) on the tastes of the user that do not necessarily have to be true, which can lead to give a wrong suggestion. And therefore, does not meet user requirements.

**Explicit Feedback**

Explicit feedback is based on the direct and deliberate action of the user to indicate those objects of the system that interest him. This action can be achieved by means of numerical votes or, simply, indicating whether or not the object is to the liking of the user. This type of feedback also presents problems, such as the willingness of the client or the time invested in it or even the veracity of the information entered into the system by the user.

**Real versus Synthetic Data**

Another interesting question is to choose a set of real data (compiled from real users on real objects) or a set of synthesized data (with no real basis, specifically created for the Recommendation System). The latter are easier to obtain, since we avoid having to conduct surveys or other methods of collecting real information, although they are only used in the early stages of system development, and then replaced by real data once it has been accumulated enough information.

**Online Analysis versus Offline Analysis**

It is important to decide whether we are going to work on data online or offline. In the offline analysis, a technique or filtering algorithm is used to make predictions about the dataset, evaluating the results of those predictions using one or several error metrics. This type of analysis has the advantage of being fast and economical, but it presents two important drawbacks: the problem of data scarcity and the problem of obtaining the goodness of the prediction as the only result.

On the contrary, the online analysis allows to obtain more results, among which stand out the performance of the participating users, the satisfaction of the same, etc. However, it turns out to be slower and more expensive than offline analysis.

3.3 Collaborative Recommendation Systems

Collaborative Recommendation Systems are those that make recommendations based solely on terms of similarity among users, that is, they combine the valuations of the objects, identify the common tastes among users based on such valuations and then recommend objects that are of the taste of other users of similar tastes to the current user.

The techniques for developing the first Recommendation Systems for collaborative filtering were based on methods from data mining. For this, it was distinguished between a
learning phase (offline) in which the model is learned, as it happens in data mining, and a recommendation phase (online) in which the model obtained from the phase prior to a real-life problem, thus producing recommendations for system users. However, currently this type of technique is not usually used, because the interaction of users with the system, it is more convenient to use a relaxed learning paradigm (the model is built and updated during system operation).

The theoretical basis of the Recommendation Systems is simple: there are groups of users, who will be those who maintain similar profiles, and a user of a group is recommended objects that have not yet experienced, but have experienced and valued positively other users of your “group”.

**Phases of SR Collaborative**

![Figure 2 Schematic of a Collaborative Recommendation System](image)

Currently, three fundamental stages in the operation of all Collaborative Recommendation System:

1. The system saves a profile of each user, which consists of evaluations of objects known by him and belonging to the database on which will be worked.
2. Based on these profiles, the degree of similarity between users of the system is measured and user groups with similar characteristics are created.
3. The system uses the information obtained in the previous two steps to calculate the predictions. Each user will be advised of objects that have not been previously evaluated and that have obtained the highest values for that prediction.

**Measures and Techniques Used in Collaborative Filtering Algorithms**

The first step in making these recommendations is to form groups with the users or items most similar to each other. For this purpose, we use similarity measures and a K-nn classification algorithm. Then, a prediction technique will be used to estimate the user’s valuation on certain items. In this project the cosine coefficient has been chosen as a measure of similarity for the implementation of the collaborative filtering algorithm.

**Notation**

Before enumerating the formulas that will see next, it is convenient to make clear the notation that is going to be used to not cause confusion to the reader:

- A user is that element represented by $u_1 \in U = \{u_1, u_2, \ldots, u_n\}$
- An item will be that element represented by $i \in I = \{i_1, i_2, \ldots, i_n\}$
- The similarity between two items $i_j$ and $i_k$ will be given by $s(i_j, i_k)$
- An assessment of a user $u_i$ on an item $i_j$ will be given by as $r_{u_i, i_j}$
- A prediction of a user $u_i$ about the item $i_j$ will be represented as $P_{u_i, i_j}$

**Algorithm K-nn**

A crucial step in the realization of a quality Collaborative SR is the formation of user groups (if it is a memory-based Collaborative SR) or of items (if it is based on models, as in this project) of similar characteristics. This activity is part of what we know as classification problems, and there are several techniques, called classifiers, to solve this problem. One of the most widespread classifiers is the K-nn algorithm, which will be used in this project to form the most similar group of items for each of the items in the database.

In more detail the algorithm K-nn consists of:

**Operation:** Being the element or object to be classified, you must select the $k$ elements with $K = \{i_1, \ldots, i_k\}$ such that there is no example $i'$ outside $K$ with $d(i, i') < d(i, i_j)$, $j = 1, \ldots, k$. Once the neighbors are found, one can proceed to the classification of two possible forms: Vote by the majority: The new object is classified according to the predominant class in the objects of $K$.

Vote compensated by distance: The object is classified based on its distance weighted with the rest of objects of $K$.

**Description of the algorithm:**

A new object appears $i_a$. We obtain the $k$ objects of the set $E_{closest}$ to $i_a$. The object is classified $i_a$. Of one of the two forms mentioned above.

**Main features:** It is a robust algorithm against noise when $k$ is moderate ($k > 1$). It is very effective when the number of possible classes is high and when the data are heterogeneous or diffuse. It has an order of complexity of $O(dn^2)$, being $O(d)$ the complexity of the metric distance used. The fact of not using models but the entire database causes it to be inefficient in memory. It serves both for classification and numerical prediction.

**Similarity Measures**
To establish the similarity between objects we must define a measure that allows us to evaluate the degree of similarity between them. It is important to note that in this project, to calculate the similarity between two items x and y, only those users who have evaluated both items will be taken into account, not being taken into account the rest.

There are many measures of similarity in the literature, of all, we will review two of the most used.

Pearson’s Correlation Coefficient: This coefficient is an index that measures the linear relationship between two quantitative variables, being independent of the scale of measurement of these variables and giving a result within the interval [-1,1]. It is calculated by the following expression:

\[ S(x,y) = \frac{\sum_{i=1}^{n} (x_i - \bar{x}) (y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2 \sqrt{\sum_{i=1}^{n} (y_i - \bar{y})^2}} \]

Being \( \bar{x} \) e \( \bar{y} \) The mean of x and y, respectively.

Cosine Coefficient: This method assumes that two items are represented by vectors in space, so the similarity between them will be given by the cosine of the angle they form. The expression for its calculation is the following:

\[ S(x,y) = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} (x_i)^2 \sqrt{\sum_{i=1}^{n} (y_i)^2}} \]

Being \( x_i \) the value of the object x for the user i, \( y_i \) the value of the object y for the user i and n the number of users who have evaluated both x and y.

**Prediction Algorithms**

After calculating the set of neighbors for each item, we must combine the valuations of this set to perform the user’s prediction on that item. Choosing the right technique to perform the prediction is the most crucial step of collaborative filtering.

Choosing one prediction algorithm or another depends on the nature of the data set, since each algorithm best fits a specific data set. In our case, we will use the technique called weighted sum. This method calculates the prediction of an item given by:

\[ p(u, i) = \frac{\sum_{h=1}^{k} s(i, a_h) \cdot ru_{u, a_h}}{\sum_{h=1}^{k} s(i, a_h)} \]

K being the k most similar items to \( i_u \). This technique tries to capture how the active user evaluates items similar to the one he wants to predict. To ensure that the prediction falls within the previously defined range, it is necessary to weight these assessments with similarity.

**4. Software Engineering**

The project that has been carried out consists of the development of an application of an Internet recommendation system based on a hybrid collaborative filtering algorithm, with a context-like algorithm to recommend songs to the users, following a client / server. This music recommendation application consists of three fundamental components, which are the four basic pillars of the development of the project:

- A database, which includes all data related to system users, songs, user ratings on songs, predictions.
- An application interface, in the form of an Android application, from which users will access the system, listen to and evaluate songs, and receive music recommendations as playlists.
- A collaborative filtering algorithm in conjunction with a context-like algorithm, which is responsible for calculating predictions from the database scores, users and songs in the database.

**Functional requirements**

Being in front of a project of academic type, no client is available to obtain the requirements, reason why we based on other Systems of Recommendation of music existing in the market, of recognized success, to establish the requirements.

The functional requirements will be:

- Logging into the system: The system must provide a mechanism so that the user can register in the system, in order to offer a personalized service.
- View Menu Personalized Songs: The system must be able to offer a series of songs to the user based on the tastes and preferences indicated at the time of registration.
- See Menu Recommended Songs: The system must be able to offer songs in its entirety, either by collaborative filter or based on choices. Mark songs as favorites: The system must allow the user to indicate songs as favorites or as non-favorites if they already are.
- Updating the recommendations model: The system must update its recommendation model from time to time, incorporating the new ratings that users have made on songs.

**Non-Functional Requirements**

Since our project is based on a client / server architecture, it will be necessary to distinguish between the requirements of the client computer and the requirements of the server. A part will
need a development computer of similar characteristics to the server equipment. The customer's computer equipment requirements are quite simple. Only a mobile device with the Android 4.1 operating system, with Internet connection (preferably Wifi, although it can be used with the data connection) is necessary.

Since this is supported by Cloud services, the Server Computer requirements are a basic SQL account in Microsoft Cloud Services, Azure Services.

**Interface Requirements**

The requirements of the graphical interface between the application and the user are closely linked to usability and its principles [10]. The basic principles of usability, which will be associated with the non-functional requirements that the graphical interface must meet:

- **Ease of learning**: refers to those features of the interface that allow new users to understand how to use it initially and how to obtain a maximum degree of productivity.
- **Flexibility**: This principle of usability states that there must be several ways in which the system and the user exchange the information.
- **Robustness**: It is the level of reliability of the system, or the degree to which the system is able to tolerate failures during the interaction.

**System Analysis**

Once it is known, the purpose of the software project, its properties and the constraints to which it must submit, is the time to analyze the system and create a model of the system that is correct, complete, consistent, clear and verifiable. For this, the use cases will be defined according to the previously obtained requirements and, afterwards, the main scenarios and event flows of these use cases will be described [11].

**General Model**

The android application will connect to the Last.Fm API to get information about the user and their preferences. This in turn will receive information from the application when creating a new user who does not have a profile in Last.Fm. The application additionally each record and recommendation that is created, will leave a record in the database that is stored in the Azure cloud service. The database will provide song information and user preferences to more accurately use Last.Fm's recommendation algorithms to achieve greater accuracy at the time of recommendation.

**Programming languages used**

**Customer Part**

*Java*: Object-oriented language whose potential resides in the compilation to intermediate bytecode of the applications developed in this language. This allows an application to run on multiple platforms and independently to hardware through a Java virtual machine. Java is the language used to develop the logic of applications for Android.

*XML*: Markup language for documents that allows you to store information in a structured way. Android uses it as a standard for the development of user interfaces.

**Server Part**

*Last.fm API*: The Last.fm API allows calling methods that XML responds in REST type expressions. In general terms, a method is sent in conjunction with specific arguments to the root URL. The API supports multiple transport formats but always takes XML by default.

*SQL*: It is a database access language that exploits the flexibility and power of relational systems and allows a wide variety of operations. It is a "high level" or "non-procedural" declarative language that, thanks to its strong theoretical basis and its orientation to the management of recordsets (not to individual records) allows high productivity in coding and object orientation. In this way, a single statement can be equivalent to one or more programs that would be used in a low-level register-oriented language.

**Development tools**

This project has implemented two parts, one client and another server, and each of them, for language needs, has needed different development tools.

The client side, which consists of an Android application and its programming language is Java, has been implemented in the Android Studio development environment. Android Studio is an integrated development environment for the Android platform. It is based on JetBrains IntelliJ IDEA software, and is published for free through the Apache 2.0 License.

On the other hand, the server side, it was necessary to create the necessary statements to create the database, so we used Microsoft SQL Server Express version 2012.
The development language used (by command line or through the Management Studio graphical interface) is Transact-SQL (TSQL), an implementation of the SQL ANSI standard, used to manipulate and retrieve data (DML), create tables and define Relationships between them (DDL).

### Designing the data

The structure of each of the information elements of the system was determined, that is, the structure of the data on which to work. These elements are:

- The songs, of which we know the id of the song in Last.Fm, the name of the artist or interpreter of the song, the musical genre to which the song belongs.
- The users, of whom we know their identifier, name, password and e-mail address.

Once the information elements of the system determined, their representations must be obtained in the form of tables from a database. To do this, a conceptual design of the database must first be performed to obtain the required tables. To implement this conceptual design, the Entity-Relationship model (E-R) will be used.

### Conceptual scheme

We need to turn our information elements into entities or relationships. In our case, Songs and Users and accounts of Last.Fm will become entities of our conceptual scheme, and Valuations becomes relationships that join the entity Songs with Users.

We can determine a total of three tables in the database, taking into account that each entity of the ECM is transformed into a table and the attributes of an entity are converted into the fields of the respective tables.

Therefore, according to the ECM, we will obtain the following tables:

**USER**

It is a table that contains as many rows as users are registered in the service and have the following attributes:

- **UID**: Integer. Primary key. Numeric identifier and univoco of the user.
- **LID**: Integer. Foreign key. Identifier that relates the user to his Last.Fm account.
- **NAME**: String of 60 characters. User name.
- **KEY**: 32-character string. User password encoded using an encryption algorithm.
- **MAIL**: String of 64 characters. User's e-mail address.

**LASTFM**

Table containing the following fields:

- **LID**: Integer. Primary key.
- **USERNAME**: Text string. Name of the user in platform Last.Fm.
- **USERPASS**: Text string. User password on Last.Fm platform.

**SONG**

It is a table that contains as many rows as songs are registered in the service and have the following attributes:

- **CID**: Integer. Primary key. Numeric identifier and univoco of the song.
- **NAME**: Text string. Name of the recommended song.
- **ARTIST**: Text string. Author of the song.
- **ALBUM**: Text string. Album to which the song belongs.
- **GENDER**: Text string. Genre to which the song belongs. You can take the following values: o Metal, disco, world, poprock, jazz, indie, classical, reggae, techno, industrial, pop, hiphop, lounge, country, latin, rock, electronic, dance.

There is, however, one exception. Since we are going to use the Recommendation library [11] it is necessary to add several more tables to get the recommendations. Specifically they are:

- Table **track_getinfo** (song features).
- Table **track_getdown** (song predictions).
- **User_getinfo** table (user characteristics).
- **User_getfriends** table (recommendations for users).

### Results

For this test, a Google Nexus 5 Smartphone (API Level 21) was used. The application called MusicRecommender was run and a user named "eluntux" with key "2141" was created. In addition, the Rock and Metal styles were selected by way of test, so that the system delivered an initial recommendation. When the created user enters the system it appears a list with recommendations of which you can see in detail each song.

![Figure 4 MusicRecommender Main Screen](image-url)
6. Conclusions

It has developed an Android application of a Recommendation System capable of providing users with songs that, without being evaluated by the user, are estimated to be to their liking. For this, the application has been developed around a Collaborative Recommendation System, following the whole system as a whole client/server architecture.

From the very beginning of the conception of the project, the intention was to create a service that would allow any type of application to access the musical recommendation system. In this way different users from different systems register in it, listen to different songs from different artists and musical genres and perform a simple evaluation on the songs. Based on these preferences, the Recommendation System creates a user profile and offers the user a series of recommended songs according to the tastes of the user and other users of similar tastes.

For the realization of the project we have compiled a set of musical data. For this, we have chosen to generate a database of real music albums, so that the prototype version of our system has 1652 songs from 5 different musical genres. The server does not host any music files at any time; they are requested directly from the music database.

First, the properties that the system was to satisfy, as well as the constraints to which it was subjected, were determined. Next, a correct, complete, consistent, clear and verifiable system model has been created. Finally, this model has been codified in a prototype version and installed on the server.

7. References