Abstract

The Intelligent Jukebox uses Artificial intelligence and feedback to choose music that a person would like to here. It models listeners of music with three categories of attributes: preference for a song, artist or genre, (either overall or at a specific time of day), the number of times a song has been played, and the time since the listener last heard a song. It plays music while at the same time determining what music they prefer. The system bases all of its decisions on only positive and negative input from the user while a song is playing. By using this feedback, the system builds up a model of what a person likes to listen to and when they like to listen to it. It creates a listening experience designed for the user relative to style of listener and style of music. Using an artificial intelligence model, this system can use a representation of several different approaches to listening to music to replace direct manipulation in the context of listening to music from a jukebox or radio.

1. Introduction

The history of selecting music with AI systems is large. Ringo, written by Shardanand U. and Maes P [1] was an early collaborative filtering system that selected music based on what your friends liked. Even the creation of computer music through artificial intelligence has been approached, including Jim Davis's (about 1987 media lab) thesis, in which a system literally creates music based on models of kinds of music to make a jazz player. IBM's Kid Riffs, written by David Jameson, is another such system that plays music with a person, allowing them to intrude as much as they can and be as effective as they would like. The music research museum in Seattle has many demonstrations in which collaborative music making and listening are focused. This paper takes music listening in a car or in a house a step in another direction. Our goal is to understand the difference between kinds of listeners and kinds of music while only using positive and negative feedback. The analogy is similar to kicking a jukebox or putting coins into it, and it is the goal of this system to make that adequate enough to drive the music listening experience.

Every year many people have automobile accidents while tuning their radio in a car. Many say that running the controls of the radio is more complex and dangerous than listening to a cell phone while driving. [2] Our approach is to make a corpus of music that has tags on it along with an agent-based architecture with different voters making decisions by helping contribute what they believe about the meaning of feedback, or lack of feedback, from a user. In particular, a list of agents that act as voters has been created. The list of agents notice qualities about the corpus of music, such as what in the database is least recently played, most recently played, the frequency they are played, which are least played and most played, and highest overall ranking as well as highest rating for this time of the day. In addition, the jukebox decides if it is choosing by song, by artist, or by genre. It can make decisions based on any of them. As well, the jukebox attempts to categorize the user. The kinds of users that we expect to hear are users that like to listen to the same song over and over again, users that never like to listen to the same kind of music over and over again, people that like to listen to music as a background, or people that like to listen to it as an engaging activity. By selecting the kind of listener model that a person has, we are able to select what kind of music they want.

2. A Precursor to the Intelligent Jukebox: The Multimedia Bed

Prior to the development of the Intelligent Jukebox, I worked in the Context-Aware-Computing lab on the Multimedia Bed. The majority of my work on the bed was focused on eye-tracking and was unrelated to the jukebox. However, one aspect of the bed was a primitive jukebox which responded to feedback. This jukebox already had the idea of feedback driven creation of a music user model. [2] The original intent was to develop a jukebox that could be easily controlled using a high eye blink rate as a negative feedback staring as a positive feedback. [3] This meant that there had to be as little different kinds of input as possible. In the bed system, the user could turn the jukebox on and off and tell it if it did not like a song all through eye movement. However, the mechanisms this jukebox used to pick a song for the user were very simple. Each song was given a rating, which was increased if it was listened to for a certain period of time, and decreased if the user asked for a new song before that time threshold. The next song played would simply be the song with the highest rating.

The system would simply not know enough about what songs a person liked and why. In addition to being incapable of accurately representing a realistic user model, it would have been very difficult to make this system work with a large database of music. Because of this, I decided to take on the task of expanding on the idea of the Intelligent Jukebox, with the idea that it could be applied to other mediums than the bed. For instance, this type of interface could be quite useful in a driving situation. Many needless accidents are caused by drivers who are paying attention to their stereo system, trying to find music that they want to hear. A system which would require minimal feedback to play music the listener desires would allow him or her to focus on the task of driving.

3. Expanding the Jukebox: Different User Models

In order to make the Jukebox more robust, the first task was to think about *how* people listen to music. In doing this, it became clear that different people listen to music in different ways. For example, one person might like to listen to one type of music in the morning and another in the evening. Another might want to listen to only their favorite artist. Still another might be interested in hearing a wide variety of music, and get bored with hearing the same things over and over, while someone else actually prefers to listen to a few songs over and over again. Because there are so many different kinds of listeners, it was my goal to develop a system which could model several of them. It was clear that this would involve more than a simple rating system, as it would be necessary to take into account time of day, the time since something was played last, the number of times it was played, and of course some sort of rating system which would determine how much a listener "liked" a particular song. The goal of the system is not only to determine *what* a user likes, but *when* they like to listen to it and how *often*.

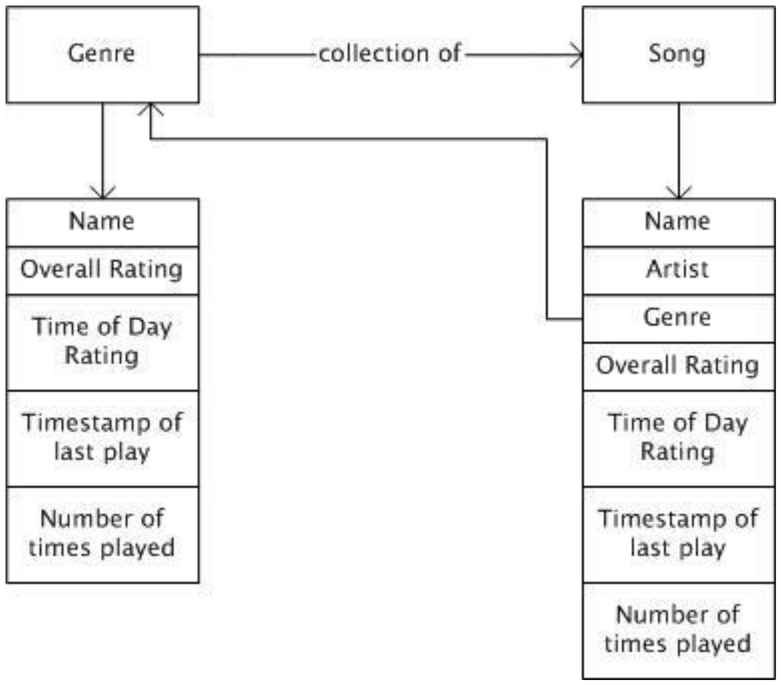
4. Implementation

The original jukebox in the Multimedia Bed was implemented in Macromedia Director's proprietary language, Lingo. However, it was decided that project should be moved into more flexible format, so this new version of the jukebox was written entirely in Java. Not only does the language offer more flexibility, because it is platform independent, it would be simple to create different interfaces for the system, such as an applet on a web page, a GUI interface, or even the knobs on car stereo.

4.1 Songs and Genres

Each song in the database is represented by a Song object. This object keeps track of several different features of the song, such as its current rating, the rating for it at different times of the day, the number of times it has been played, the last time it was played, the artist of the song and the genre the song is in. The Song object implements a thread interface, so that it may play songs directly by calling the objects play() method. There is also a stop() method to stop a song before it ends.

Figure 1: Genres and Songs



A Genre object is much simpler. Its main function is as a container for several Song objects. It also has ratings, time stamps, and frequency of play information associated with it, much like the song objects. The ratings and frequency of play records are cumulative of all the ratings and frequency records of all the songs in the genre, while the timestamp is that of the most recently played song in the genre.

4.2 Voters

The Jukebox needed a way to determine which song would be played next. The system that was devised included several Java Objects called Voters. Each Voter picks a song based on a distinct user model. These listener model voters represent a different “kind” of listener, such as those described briefly above. There are six different Agent voters in the system. They are as follows:

- **High Rating Voter** – picks a song based on the overall rating.
- **Time of Day Voter** – picks a song based on the rating for that particular time of day.
- **Most Played Voter** – picks based on the number of times played.
- **Least Played Voter** – picks based on the number of times played.
- **Most Recently Played Voter** – picks based on a timestamp of the last time something was played.
- **Least Recently Played Voter** – picks based on a timestamp of the last time something was played.

The model of the user is the settings these voters have for the situation with them. The system must decide which one of these Voters is the winner so that it knows which results to use. For this reason, each Voter also has a heuristic value, or rating, associated with it. The Voter with the highest rating wins, and its results are used.

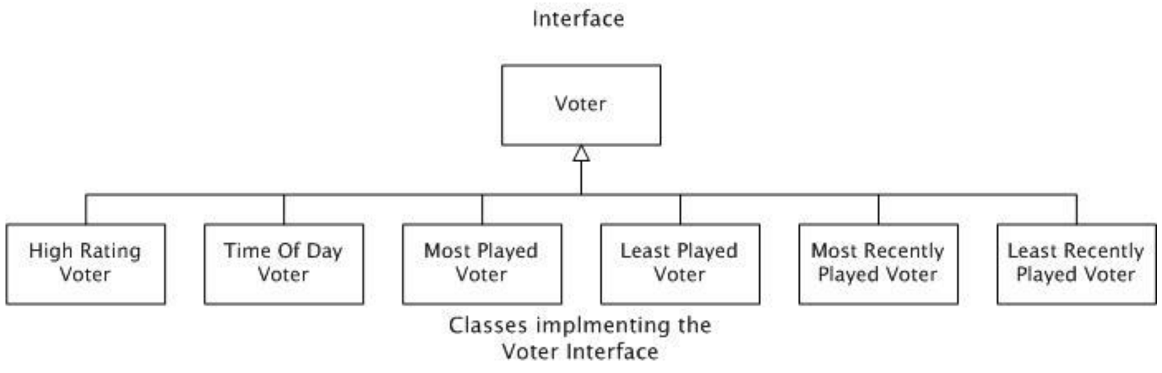

Interface
Voter
High Rating Voter
Time Of Day Voter
Most Played Voter
Least Played Voter
Most Recently Played Voter
Least Recently Played Voter
Classes implmenting the Voter Interface

Figure 2: Listener Model Voters

4.3 The idea of a "Type Voter"

This initial setup simplistically only took into account decisions based on individual songs. For this reason a second set of Voters was implemented, called "Type Voters." These voters decide whether a decision will be based made on individual songs, artists or genres. For example, let's say the Artist Voter was the type voter winner and the High Rating Voter was the winner for the listener model voters. In this case, the system would pick the *artist* with the highest rating, and choose a song from that artist. Or perhaps the type voter winner is Genre and the listener model voter winner is the Most Recently Played Voter. Here, the Jukebox would pick a song from the genre the listener is currently listening to. Currently, if the Artist or Genre voter wins, an artist or genre is picked with the listener model voter, and a random song within that subset of songs is picked. However, if the Song Voter wins, all of the songs in the database are considered by the listener model voter. We considered implementing a hierarchical system of voters that would pick songs in genres or by artists more intelligently, however that would have made the system extremely complex without a visible benefit to the user. For instance, if the Genre voter is winning, it can be assumed that the listener likes songs from the same genre, and does not necessarily care about the specific song. In such a situation, a random pick of songs in that genre is justified.

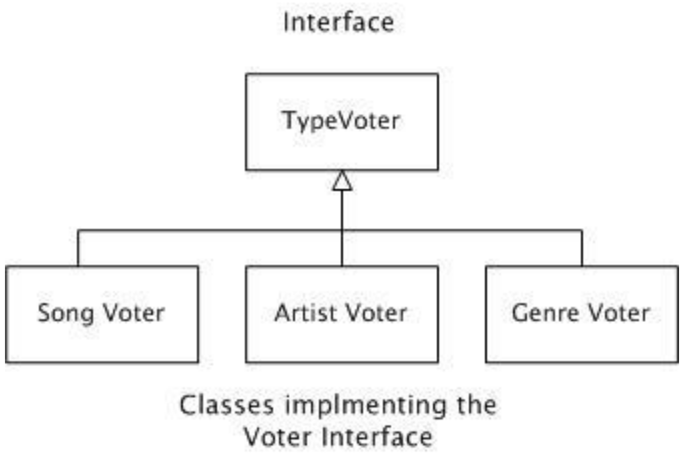
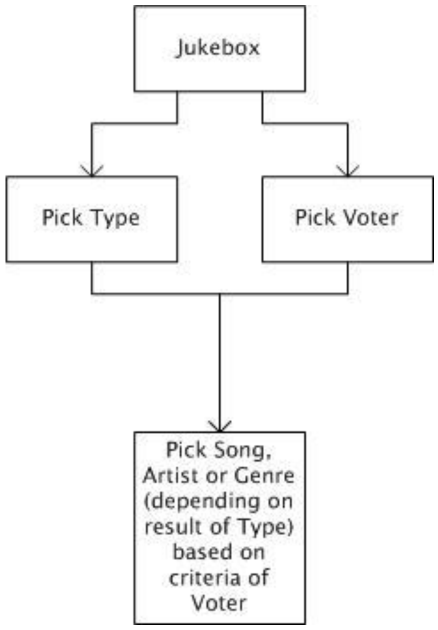


Figure 3: Type Voters

4.4 Determining Ratings: Positive and Negative Feedback alone

The system is now faced with the problem of how to determine these ratings. The solution is quite simple when using our intended interface of having only positive and negative feedback. If a song is playing and the user gives the Jukebox positive feedback, its rating will be increased. Similarly, if the user gives negative feedback, it will be decreased. In addition, the Voters must also be rated. When feedback is given, the Voter that was used to make the decision to play the current song has its rating adjusted according to that feedback. Both listener model Voters and Type Voters are adjusted after feedback is received.


Jukebox
Pick Type
Pick Voter
Pick Song, Artist or Genre (depending on result of Type) based on criteria of Voter

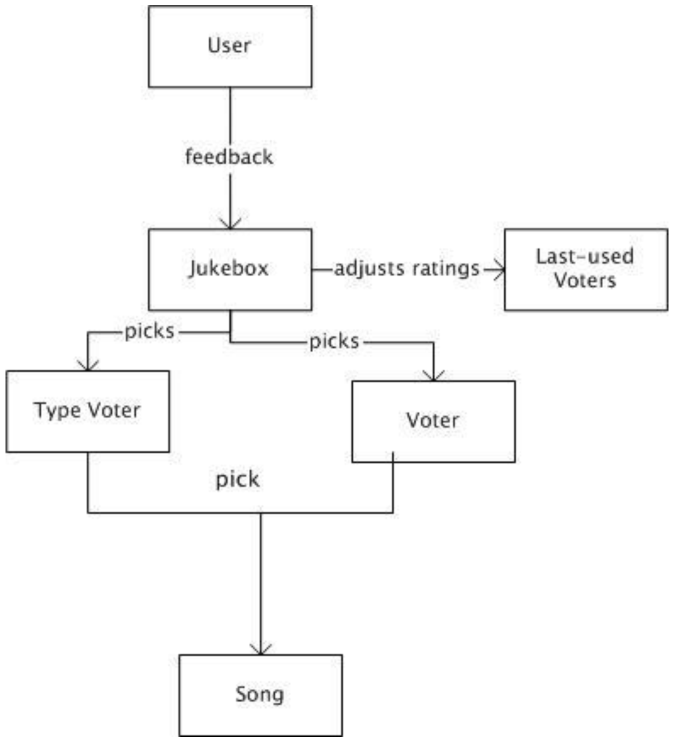


Figure 4: Flow of Control: Song Picks

Figure 5: Flow of Control: Voter Ratings

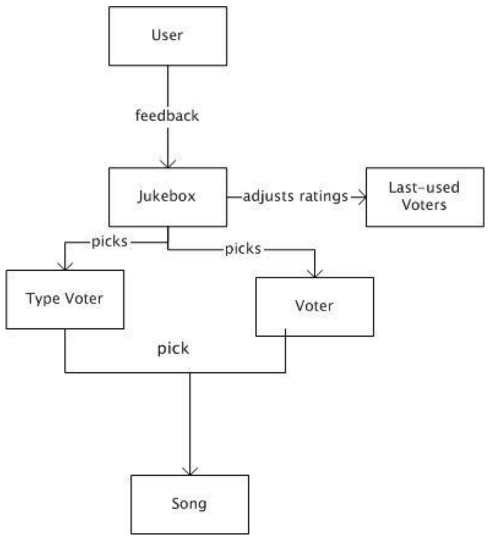


Figure 5: Flow of Control: Putting it all together

4.5 Users and the Music Database

In order to accommodate for different users, each user will have their own database of Song objects, Genre objects and Voter objects. However, different users may want to use the same database of songs. To allow for saving of information, a user's database, along with all associated ratings and timestamps, is written to a file. The music database itself must be a directory of mp3's, organized by genre and artist. When a new user is created, he or she may enter the directory of the desired mp3 directory, and the system will load all of these songs into their own Song objects, and create Genre objects for them as well as Voter objects for the user. When a user logs into the system again, the system loads this information from the user's database file, and scans the directory for new or invalid mp3's, and makes adjustments accordingly. This allows several users to use the same database of songs, while still allowing the flexibility to change that directory.

Currently, the system requires a directory structure which conforms to its standards. The music directory must contain a folder for each genre. Within each genre folder, there will be a folder for each artist, and with in each artist's folder will be the songs by that artist. The reason for implementing the music database in this format is that it allows the Jukebox to gather the important information by simply looking at the directory structure. This makes the system much easier to use because it does not require the user to enter information about each song into the Jukebox. The only requirement on the user is to put songs in the correct directory when adding them to the database.

The Jukebox is robust enough to handle cases where songs may be removed or added to the directory. Upon startup, it scans the directory for changes, and removes songs from the user’s database file if necessary, and creates new Song or Genre objects and adds them to the system when it comes across new files. This makes changing the selection of music relatively painless.

4.6 Weighted Ratings

In order to get more accuracy with ratings, the rating system was overhauled during development. Initially, the rating of a song was saved as a simple double value. However, the system was changed so that each song actually had a *matrix* of ratings associated with it. This allowed the system to determine which listener model Voter and Type Voter combination resulted in which rating. When choosing a song, the rating for a particular Voter/Type Voter is weighted more when that combination is making the decision. This resulted in greater accuracy in determining why the user gave the system positive or negative feedback.

5. Interface

The Intelligent Jukebox System was designed so that the backend of the system could easily be "plugged in" to whatever front-end interface is desired. A GUI interface was created for testing purposes, but it is really quite simple. It has a button for positive feedback and one for negative feedback ("Coin" and "Kick" respectively, in keeping with the jukebox theme). The GUI sends the positive and negative feedback messages back to the Jukebox, which performs all the necessary calculations to make its decision and play the song. This system could have just as easily been "plugged in" to the Multimedia Bed, with a front-end interface in Lingo, which would allow the positive and negative feedback to be given to the system through eye movements. Similarly, there are plans to develop an interface for a car, which would only require two buttons, which would be far less distracting for a driver than the usual stereo interface.

6. Initial Findings

During the debugging phase, we tested the system frequently and found it to be surprisingly accurate. After listening to approximately 7-10 songs and giving the system feedback on those songs, w were able to let the system run by itself for long periods of time without giving it feedback. W found that it would pick songs that we would not necessarily have picked ourselves, but found ourselves enjoying anyway. In using the system, we have discovered that rapid feedback is important. By having feedback happen after just a very small amount of music is played on a particular song, we have been able to make a system that is reactive.

7. Proposed Testing

There are a few different types of testing currently proposed for future study. In each case, the control would be a system of choosing songs similar to a radio, with up and down buttons to scan “stations” which would randomly play songs of a specific genre. Users would use each system, and the number, time, and frequency of clicks on the interface buttons (positive and negative for the jukebox and up and down for the control) would be analyzed. The different test situations would include using the system without any distractions, using the system while performing another task at the computer, and using the system in a driving situation. If the system works as we expect, we believe we will find that the user has to interface with the Jukebox less than the control system, and that the clicks are less frequent after time.

8. Conclusions and Future Work

This system demonstrates that feedback as a basis for user interface can be effective and subtle as a way of directing a computer to do things that are important on an aesthetic basis to a person. A simple representation was adequate to present information that a person would like to listen to within the way that they listen to music.

In the future, we hope to expand the system by introducing the concept of context-switching. In the current system, each user can only be modeled by one listener model Voter at a time. In addition, the system only has a short-term memory, as the most recent decisions affect decision-make the most. In addition, if a user listens to music in a certain way for a long period of time, it will take the system an equally long time to switch out of that listening model into another.

Logging of the user’s decisions has already been implemented, and the next step is to have the system analyze these logs to determine the most frequently used user models. In such a system, when a small threshold of continuous negative feedback is crossed, the system will instantly switch into another concept. This comes from the idea that different users might be different kinds of listeners at different times.

By taking this model, one could imagine making cars that do not disturb the driver while they are trying to listen to music, or other listening materials. This could be great benefit to many drivers, and could potentially save lives, as the driver can concentrate on the road instead of being distracted by their radio.

9. Bibliography

[1] Shardanand U. and Maes P. “Ringo: A social information filtering system for recommending music,” internal report, Media Laboratory, MIT, May 1994.

[2] Donald W. Reinfurt, Herman F. Huang, John R. Feaganes, and William W. Hunter. “Cell Phone Use While Driving in North Carolina,” University of North Carolina Highway Safety Research Center, 2001.

[3] Selker, T., Burleson, W. “Context-Aware Design and Interaction in Computer Systems: Media Bed,” Proceedings of the AAAI Fall Symposium Socially Intelligent Agents -- The Human in the Loop. September, 2000.

[4] Selker, T., Burleson, W. “A Test-Bed for Intelligent Eye Research,” LREC 2002, Third International Conference on Language Resources and Evaluation, LREC Workshop on Multi-Modal Resources and Multi-Modal System Evaluation, Conference Proceedings.