massachusetts institute of technology — artificial intelligence laboratory

The Audiomomma Music Recommendation System

Mariano Alvira, Jim Paris and Ryan Rifkin

AI Memo 2001-012 July 2001
CBCL Memo 199

Abstract

We design and implement a system that recommends musicians to listeners. The basic idea is to keep track of what artists a user listens to, to find other users with similar tastes, and to recommend other artists that these similar listeners enjoy. The system utilizes a client-server architecture, a web-based interface, and an SQL database to store and process information. We describe Audiomomma-0.3, a proof-of-concept implementation of the above ideas.

This report describes research done within the Center for Biological & Computational Learning in the Department of Brain & Cognitive Sciences and in the Artificial Intelligence Laboratory at the Massachusetts Institute of Technology.

This research was sponsored by grants from: Office of Naval Research under contract No. N00014-93-1-3085, Office of Naval Research (DARPA) under contract No. N00014-00-1-0907, National Science Foundation (ITR) under contract No. IIS-0085836, National Science Foundation (KDI) under contract No. DMS-9872936, and National Science Foundation under contract No. IIS-9800032.

Additional support was provided by: Central Research Institute of Electric Power Industry, Center for e-Business (MIT), Eastman Kodak Company, DaimlerChrysler AG, Compaq, Honda R&D Co., Ltd., Komatsu Ltd., Merrill-Lynch, NEC Fund, Nippon Telegraph & Telephone, Siemens Corporate Research, Inc., and The Whitaker Foundation.

1 Introduction

For a variety of cultural and technological reasons, downloading music from the Internet is a new national pastime. Primary enabling technologies for this "revolution" include the development and widespread use of the MP3 file format (mpeg level 3 encoding), the expansion of fast networks throughout universities, and the development of easy to use file distribution software (such as Napster). The MP3 file format allows music to be compressed to approximately 1 Megabyte per minute of music, while retaining near-CD quality. High speed networks allow, in principle, the fast transfer of music files between users, and music sharing systems such as Napster allow users to share their music with others. As a result, many people now find most of their 40 Gigabyte hard drives accomodating their MP3 collections.

Although it is easier than ever to find and obtain music we are looking for, a new problem presents itself: how do we find *good* music? Previous approaches to this problem include Usenet discussion groups, Napster chat channels, and user rating systems. The first two approaches involve the user actively interacting with other users. The third approach requires that the user actively rate content. All three introduce additional burden on the user, and have not to date led to widely adopted systems.

Audiomomma is a system designed to solve this problem differently. Audiomomma consists of a client-side system that observes what artists a user listens to, and stores this information in a central database. Audiomomma users are then matched up to other users with similar music taste based on music they play on their computer. Audiomomma provides an easy interface for suggesting to its users music artists they will probably like. The system was designed from the beginning to be computationally efficient, scaling to large numbers of users.

Essentially, by observing what artists a user enjoys and storing this information automatically, Audiomomma acts as a substitute for manual user ratings. Any algorithm which takes as input user rating data could be integrated into the Audiomomma system. This removes the burden of active rating from the user, resulting in an easy-to-use system.

A proof of concept system Audiomomma-0.3 was created at the Center for Biological and Computation Learning at MIT. This paper describes the construction and performance of this implementation of an Audiomomma system. The system can be used via http://audiomomma.mit.edu.

2 System Architecture

The data-paths in Audiomomma-0.3 are structured as shown in figure 1. Execution of the system can be broken into four segments:

1. User creates an account.

2. User plays an MP3.

3. Audiomomma crunches profile data.

4. User gets artist suggestions.

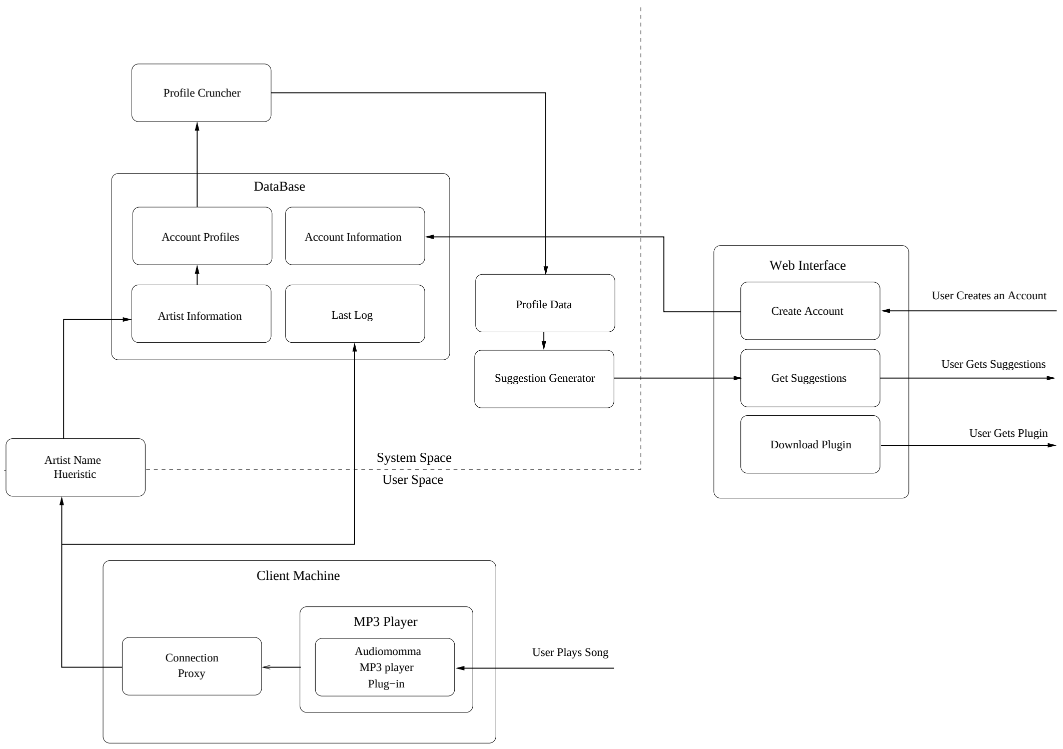


Figure 1: Audiomomma-0.3 data paths

2.1 User Creates an Account

A user creates an Audiomomma account by visiting the web-site. They submit to Audiomomma, through a HTML form, their first name, last name, and email address.

The new user information from the HTML form is entered into the "Accounts" table in Audiomomma's database (see Table 1). The new user is assigned a unique User ID number (UID) and random password.

Audiomomma emails the new user a cordial greeting to the system; it includes simple instructions as well as his password – serving to welcome the new user and to verify his email address.

Accounts	
email	*unique*
first name	
last name	
password	
AID	*unique, numeric*

Table 1: Accounts Table in Audiomomma-0.3 database.

2.2 User Plays MP3

After the user has created an account with Audiomomma, he must download the client software which includes a plugin for his MP3 player.

Each time the user plays an MP3 the client software attempts to send the artist name stored in the ID3 tag of the MP3. If there is no ID3 tag the client sends the filename of the MP3. Audiomomma-0.3 uses a plug-in for Nullsoft's WinAmp as well as a stand-alone program which handles sending the HTTP requests to the server. For complete details of the client implementation used in Audiomomma-0.3 see Appendix A: An Audiomomma Client for Windows

The data sent to the server is the user's email address, password, some number of filenames, the same number of artist names (from the ID3 tags), and a count equal to the number of filenames/artists pairs sent in the request. If the ID3 tag for a particular MP3 is not present, the artist field is left blank.

This protocol allows for multiple filenames/artists to be sent with a single request. Support for this is necessary for people who do not have a persistent Internet connection. In their case, the client caches their songs played until a connection is available.

If there is no ID3 tag for a particular request (i.e. the artist field is blank) then the artist is extracted from the filename. This is possible since most MP3s are named according to some logical scheme. A common one, for example, is "Artist Name - song title.mp3". Audiomomma-0.3 contains a sophisticated set of patterns and filters for reducing a MP3 filename into something that is probably an artist name. The artist names from ID3 tags are similarly processed to normalize them (e.g. uppercase characters are made lower and spaces are removed). We call the reduced filename and normalized artist name 'squash'.

For a complete discussion about the details of the squashing heuristic see Appendix B: Obtaining and Normalizing Artist Names from MP3 Filenames.

Audiomomma assigns each squash an Artist ID number (AID) and keeps track of a human-readable form of the artist name. This is done in the Artist Information table of the database. See Table 2.

Artist Information	
AID	*unique, numeric*
artist name	
squash	*unique*

Table 2: Artist Information Table in Audiomomma-0.3 database.

Some typical entries may look like:

aid	name	squash
14	The Police	police
15	John Mellencamp	johnmellencamp
16	Bush	bush
17	Soul Coughing	soulcoughing
18	Ben Folds Five	benfoldsfive
19	R.E.M.	rem

New entries are created for squashes that don't exist in the database already.

When an HTTP request containing user listening information is received, Audiomomma uses the Artist Information table to lookup or create an AID for each squash not already in the database. The UID of the user is obtained from the Accounts table with the supplied email address and the password is verified. Then a counter in the "Profiles" table (see Table 3) of the database, which indexed by UID and AID, is incremented.

Profiles	
UID	*numeric*
AID	*numeric*
count	*numeric*

Table 3: Profiles Table in Audiomomma-0.3 database.

It is in this way Audiomomma keeps track of the number of times each user plays each artist. It is easy to perform numeric calculations on the data in this format. The profiles table can be linked to human-readable data by table joins with Artist Information and Accounts tables.

2.3 Audiomomma Crunches Profile Data

Audiomomma is designed to be scalable to the order of millions of users. Millions of database lookups and preference algorithms over millions of points are expensive in terms of time. This creates a serious design problem when the lag time from when a user requests a suggestion and his suggestion is actually computed is considered. Ideally computation should be on the order of seconds.

A suggestion generator can do two types of precomputation to reduce the lag time:

1. crunching database lookups into a pregenerated format that can be quickly read by a preference algorithm.

2. precompute costly operations of the preference algorithm.

Both of these solutions create an 'off-line' system, that is, a system whose suggestions will not be based on real time data. This is problematic, in that we would like our system to give as up-to-date as possible recommendations. Audiomomma handles this by using real-time data for the requesting user's profile data, but older data for finding users that closely match the current user. In other words, if I am an Audiomomma, I will find other users whose profiles at one point matched my current profile, although I may not necessarily match their *current* profiles. Currently, the "other users" portion of the matching algorithm (see below) is updated every night, so the users matched against can be up to one day out of date.

This approach, combined with the fact that all changes to user preferences are stored in the database, creates some interesting features of the system. One user specified option could be to match the user up with other users from five minutes ago, one month ago, or a decade ago. This could be a way for people in the future to get not just popular songs of today but music that was popular years ago.

Audiomomma-0.3 uses option 1 only. Option 2 is not used since the underlying suggestion algorithm in Audiomomma-0.3 is k-nearest neighbors which has no reasonable precomputation that can be done over millions of points.

2.3.1 Audiomomma-0.3's suggestion generator

In Audiomomma-0.3 suggestion generation occurs in two steps.

The first is crunching the profile data from the database into a format that can be read quickly. Table 4 illustrates the file format used to store the profile data. The crunched data file stores UID, AID and times_played information in a convenient format for the suggestion algorithm. Times_played is a normalized value of the count stored in the database (i.e. the current AID count divided by the total number of song plays for this UID). UID and AID are 32-bit integers and times_played is an 8-bit float. The two integers i and j represent the total number of users and the total number of artists known to the system when the file is generated. This file is generated every night, and used to find matching users whenever suggestions are requested, until a new file is made the following night.

i
UID_1
j
AID_1
$times\_played_{AID_1}$
$\vdots$
AID_j
$times\_played_{AID_j}$
UID_2
$\vdots$
UID_i
$\vdots$

Table 4: Crunched file format

2.4 User Gets Artist Suggestions

Audiomomma-0.3 uses a k-nearest neighbor algorithm to generate artist suggestions. Each UID is a point in the space with dimensions of all possible AIDs. The value of each dimension is the normalized times_played number. The Euclidean distance is found between the current user's point and all other points. Audiomomma-0.3 then goes through the points, closest points first, finding artists the current user has never listened to. Audiomomma-0.3 stops suggesting artists after some arbitrary MAX_SUGGESTIONS number is reached.

In plain English, Audiomomma finds *users* whose profile closely matches the requesting user's, then finds *artists* that those users have listened to that the current user has not yet listened to.

2.5 Conclusions and Future Directions

We have developed Audiomomma, an online, automated system for suggesting new artists to users. The system is implemented and functional. Informal experiments with toy data indicated that the system scales well to many users.

These experiments also indicated that relatively obscure but popular bands can propagate through the system: if you and I both like many of the same popular artists, and you start spending a lot of time listening to "Obscure Band X", this band will likely be recommended to me as well. There is some hope that such a system might allow people to be exposed to music they wouldn't otherwise see, and allow artists to become known to their target audience without having to be heavily marketed by record companies.

By avoiding the need for manual interaction and input, we avoid many of the problems associated with other systems for finding new music. There is hope that this system, if widely adopted, would be beneficial to many.

Unfortunately, we are faced with something of a chicken-and-egg problem. Without a lot of data, the system cannot make good recommendations, but without being able to make good recommendations, why would anyone use the system? Certainly, we urge all Windows users to go to http://audiomomma.mit.edu and try out the software. If enough people begin to use it, the recommendations should get steadily more useful. If there is enough interest, it would not be overly difficult to make additional clients, to allow Linux or Macintosh users, or users who don't play music via Winamp, to use the system.

3 Appendix A: An Audiomomma Client for Windows

3.1 Architecture

For the Audiomomma system to gain widespread use, it is important that clients capable of talking to the Audiomomma server exist for as many MP3 playing programs as possible. It is beneficial, therefore, to develop a Windows client system that makes it easy and quick to add support for new and additional MP3 programs. It was decided that an efficient method of doing this was to break the client side of the system into two parts: a standalone Audiomomma application and an MP3-program-specific plugin or helper.

The standalone Audiomomma application, which inherits the name Audiomomma, handles the details of communicating with the Audiomomma server as well as all authentication, encryption, errors, and other issues related to network communications. The program-specific plugins, on the other hand, need only to gather specific artist names from the songs that the user chooses to play and communicate these artist names back to the standalone Audiomomma application.

3.2 Communication Model

Data transfer between the plugins and the standalone application is achieved through the Windows Registry. This is not the most elegant way to transfer data, but the alternatives are also non-ideal:

- **IPC (Inter-process communication)**: Using named pipes or other offline IPC methods would be ideal, but they are only supported under Windows NT, 2000, and XP. Using Internet-domain sockets for IPC (either TCP/IP or UDP/IP) would work, but is complicated and subject to security issues when working with an Internet-connected host.

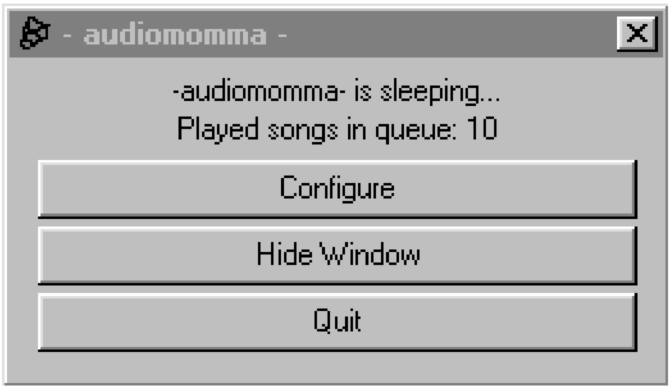
- **Temporary Files**: The artist data could also be stored in temporary files. If the temporary files are stored in a fixed location, however, the system is prone to hardware-dependent incompatibilities. If the files are stored in a user-definable location, an extra level of configuration and inconvenience is introduced to the end-user.

By storing the artist data in the registry, the plugin is kept simple (since the location in the registry is fixed), fast (since the registry is designed to be changed and indexed quickly and repeatedly), and well-supported (since all versions of Windows since the release of Windows 95 have the same registry model).

The registry is organized as a hierarchy of keys, values, and data. Upon the user's playing of a song, the MP3 program's Audiomomma plugin, helper, or native code will store information about the song in the `HKEY_LOCAL_USER\Software\Audiomomma\Plugin` key. The value is set to a filename that uniquely identifies the MP3, and the data is set to the artist name, as best determined by the plugin. One such source of the artist name is the ID3 tag of the song, but other sources such as program-specific databases or even user input could be used. If no source of this information is available, the Audiomomma server will use a heuristic on the filename in an attempt to extract the artist (as described in Appendix B).

3.3 Audiomomma Standalone Program

The standalone Audiomomma client program, as described above, deals with all of the aspects of communicating with the Audiomomma server. The client resides in the system tray in the lower right corner of the screen, using dialog boxes to interact with the user for configuration. The user-configurable options include username, password, network and proxy settings, and preferences regarding the frequency of sending artist information to the server.



-audiomomma- is sleeping...
Played songs in queue: 10

Figure 2: Audiomomma Standalone Program

Most defaults are reasonable, and users will typically only need to enter their Audiomomma username and password to get started with the client.

The client corresponding to Audiomomma 0.3 does not yet support secure connections to the server (via SSL) but is otherwise fully functional.

3.4 Winamp Plugin

The `gen_mom` Winamp plugin follows the "general purpose" plugin specification as defined at `http://www.winamp.com/nsdn/`. The Winamp plugin interface is minimal and provides no support for automatic notification of song playing, so the `gen_mom` plugin polls Winamp every five seconds to get the filename of the currently playing song. If the current song differs from the previous song, or there was no previous song, the plugin proceeds to add the song to the registry according to the model described in section 3.2. The artist name is taken from the ID3 identification tag, if one is found in the file. If no ID3 tag is found, the artist name is left blank.

4 Appendix B: Obtaining and Normalizing Artist Names from MP3 Filenames

Many MP3 files lack the ID3 identification tag, and the name of the artist is therefore unavailable. Fortunately, it is exceedingly common to include the artist in the filename of the song. The Audiomomma server uses a Perl script to apply a procedural heuristic to incoming filenames in order to best determine the artist name. A high-level overview of process is described below. Four example filenames are provided here and after each step to demonstrate the process.

```
C:\MP3s\Track_02_-_Stone_Temple_Pilots_-_Down.mp3
/mp3s/swim%20-%2001%20-%20clearview.mp3
TheyMightBeGiants-AnaNG.MP3
02-[Radiohead]-High-and-Dry.mp3
```

1. Strip directory names and file extensions:

```
Track_02_-_Stone_Temple_Pilots_-_Down
swim%20-%2001%20-%20clearview
TheyMightBeGiants-AnaNG
02-[Radiohead]-High-and-Dry
```

2. Replace URL (`%xx`) and MIME (`=xx`) encoded sequences with ASCII equivalents, and convert various forms of word delimination to spaces. The heuristic can, depending on the string, decide to use underscores, dashes, or capitalization as delimination:

```
Track 02 - Stone Temple Pilots - Down
swim - 01 - clearview
They Might Be Giants - Ana N G
02 [Radiohead] High and Dry
```

3. Strip off track numbers from the beginning, end, or middle of a line. Track numbers are identified by strings such as "track" or "trk", or by the presense of bare numbers in the string:

```
Stone Temple Pilots - Down
swim - clearview
They Might Be Giants - Ana N G
[Radiohead] High and Dry
```

4. Try to match the string to a number of predefined patterns, returning the artist:

```
Stone Temple Pilots
swim
They Might Be Giants
Radiohead
```

The examples provided here were fairly straightforward, but many steps of the heuristic are repeated, conditionally executed, or modified based on the data. As a result, much more complicated input data will pass through the heuristic successfully and return the artist.

After extraction of the artist, an additional step is taken to ensure that the artist names are consistent in the database. Articles such as "the" are stripped, artist names are converted to lowercase, and most numerals are converted to their English equivalents. In addition, names such as "Björk" are simplified to "Bjork". All of these changes are intended to facilitate later matching by ensuring that artists are referenced in a consistent way.