MusicDigger: A tool for Music Discovery

M. Adil Yalcın, Preeti Bhargava, Sravanthi Bondugula, Varun K. Nagaraja and Marcelo Velloso
Dept. of Computer Science
University of Maryland, College Park
yalcin@cs.umd.edu, prbharga@cs.umd.edu, sravb@cs.umd.edu, mvelloso@umd.edu, varun@cs.umd.edu

Abstract—There are currently many information visualization applications present for various fields that deal with large amounts of data but one such area that has often gone overlooked is the area of music. Musicians are a very collaborative group and they will very quickly generate complex networks of collaborations with other musicians and a large set of releases, also known as discographies. In conjunction with our faculty collaborator, Robert Gibson (Director, University of Maryland School of Music), we have worked to create our own information visualization tool to browse music data called MusicDigger.

MusicDigger is a visualization application that allows users to discover discographies and browse the collaborative networks of artists using the Discogs database. Discogs is a web site that contains information on artists and their releases contributed by many users who support the web site. Our tool focuses on music discovery rather than music recommendation.

I. INTRODUCTION

The current musical world we live in is a vast collection of data that is not readily available to everyone. Many professional musicians do not know exactly what is out there, because what is made more readily available are those artists who work with major labels and their major releases. There are many talented individuals who often perform under the radar and there are also many interesting collaborations that major artists and that are not as well advertised. These are the sorts of questions professional musicians would like to answer but only with the dawn of the Internet has it been possible to access so much music data.

We make extensive use of a web site called Discogs1 to access user-submitted data regarding the releases of musicians. To contribute to this site users must first find a release that does not already exist in the database and create a new entry. The users of the community then rate how correct the data is through a voting system and as long as the data is determined to be incomplete it is color coded red in the database. Through this system of community vigilance the data is determined to be correct according to the masses, much like how web sites such as wikipedia work. MusicDigger, our solution, uses the Discogs database of user-submitted data in the background and adds functionality to visualize the information. We have two major visualizations to accomplish this: a timeline visualization to view the releases of an artist/label over time and a network visualization to view the connections of a particular artist/label to other artists/labels.

Section 2 discusses previous work that has been done in visualizing music data and how our application differs from these other solutions. Section 3 discusses previous work that has been done in visualizing music data and how our application differs from these other solutions. Section 4 discusses the data we worked with from Discogs. Section 5 discusses our application in detail, the design issues we faced, and how we implemented the application. Finally, section 6 describes the user evaluation study that was conducted and section 7 presents the conclusions we gained from this study and the future work for this project.

II. RELATED WORK

There has been extensive work done in the area of visualizing music data but much of it can be tailored to two main types of tools: music recommendation systems and tools for exploring personal music collections. The tools are discussed below.

A. Music Recommendation Systems

Music recommendation systems arose because there was a market for users who were interested in listening to new music similar to artists they already liked. There are two popular web sites that do music recommendation: Last.fm2 and Pandora3.

Last.fm is a web site that uses a social networking approach to music recommendation. When a user creates their account on Last.fm they are asked to import the data of the music (scrobbling) they listen to from their computer or on internet radio and Last.fm will then find other users who listen to the same music. Once Last.fm encounters users who listen to a lot of the same songs Last.fm will recommend songs the user has not listened to yet from the play-lists of these other users. Last.fm plays these songs using the same format as radio stations, it does not allow the user to pause a song but users can skip songs they do not like.

Pandora is a web site that uses the data in the Music Genome Project to make its recommendations. The Music Genome Project has isolated over 400 different musical attributes and it tags songs and artists with these attributes. The user is then given the option to create radio stations based on a particular song/artist. Pandora finds other artists/songs that are similar

1http://www.discogs.com/
2http://www.last.fm/
3http://www.pandora.com/
to a particular song/artist and plays its recommendations to
the user like a music radio. This method of recommendation
differs from Last.fm because the Music Genome Project has
come up with musical attributes and classified songs/artists
using experts in the field of music as opposed to a social
networking approach.

B. Tools for Exploring Personal Music Collections

Tools such as MuVis [4] and MusicBox [5] allow the user
to explore their personal music collections in different ways.
MuVis focuses on visualizing a large collection of music using
treemaps. The ordering of the treemap is based on a pivot artist
selected by the user that gets displayed in the top-left corner.
Other artists in the music collection are mapped according to
their similarity to the pivot artist. The user is allowed to control
other variables such as the size and color of each node in the
treemap and the user can filter results by the duration of the
track, the release year, the genre, the beat, and the mood.

MusicBox, on the other hand, maps a music collection onto
2D space by applying principal components analysis (PCA)
to contextual and content-based features. After projecting the
songs in a collection onto 2D space, the MusicBox application
then plots a scatter diagram of the songs. The guiding intuition
in navigating this visualization is that similar songs are closer
together.

C. How MusicDigger is Different

Our tool is different from the others out there in that
it is not a recommendation system or a tool for exploring
personal music collections. Our tool focuses on allowing users
to discover music through the connections of artists. In that
sense our tool allows users to explore the music collection of
the Discogs database.

III. DATA DESCRIPTION

As mentioned earlier, we have made extensive use of data
found on the Discogs web site. Discogs identifies the data
under three main categories - Artists, Labels and Releases.
Artists can be individuals (Ex: Eric Clapton) or groups of
artists (Ex: Beatles, Pink Floyd) who have a set of releases
which is referred to as discographies. Each recording can
be released on many formats: Vinyl, CD, DVD, etc. Other
information included with the release is the date of production,
the country of production, the genre and styles, the credits, and
the track list. The genre and style determine what category the
music falls under: rock, pop, jazz, etc. The track list shows
the number of songs recorded for a particular release and the
credits show who contributed what to each song.

Record Labels (Ex: Sony Music) are the companies that deal
with marketing releases, they can be used to categorize music
into styles and artist communities, especially for independent
labels. A label also has a list of albums released within it, and
also can include sub / parent labels.

Discogs allows users to access monthly data dumps of the
entire Discogs database\(^4\). The data includes all the releases

\(^4\)http://www.discogs.com/data/

users have contributed to the database when the dump was
made and it is provided in the XML format specified by the
Discogs API. The Discogs API allows users to write code to
remotely request the data of certain releases if the release id
of the Discogs database is known. We have made use of data
from the latest database dump made on April 2011.

IV. MUSICDIGGER - DESIGN AND IMPLEMENTATION

MusicDigger is an application written in C++ that allows
users to discover discographies and browse the networks
of artists using the Discogs database. Our code has many
moving parts but the main two visualizations are the timeline
visualization and the network visualization.

1) Timeline visualization: The releases of an artist or a label
are displayed on a timeline making it easier to visualize
along with various other filtering and view controls.

2) Network visualization: Connections are identified be-
tween a artist of choice to other artists based on col-
laborations in releases, links through other artists/labels
etc.

In order to feed the data to these two visualizations we
required code to parse XML files, populate our own database
using Discogs data, write interfacing functions to query the
database and parse the retrieved information, and code a
module to make release and image requests in the background
so we would not have to store all that data on a users machine.

A. XML Parser

The XML data available at discogs is required to be parsed
to extract necessary fields used in the application. These fields
are mentioned in Appendix A. We use a stream based parser,
Expat to parse the XML sequentially as the huge size of
the data disallows the use of a traditional tree based parser.
Chunks of data is sent to the parser that stores the state of
the parse elements. This state preserved by the parser enables
the parsing of sequential buffered data. The data stream is
encoded in UTF-8 format and any illegal character is skipped
by the parser. Different schema’s present with the label, artist
and release data has been incorporated into the parser. The
parser is analogous to a plug-in service. It can be used by
any application which provides an XML and the type of the
data. As each structure is parsed, the database populates the
 corresponding tables. Also, the release data loaded in the
memory by the DataMiner uses the Parser to extract release
track information.

B. Database and Interface

The database was designed to consist a sufficient number
of tables which would support easy data retrieval for various
features implemented in the application. The database schema
used for the application can be found in Appendix A. The data
parsed from the XMLs was populated in an SQL database
using the SQLite tool which provided versatile features to
handle the large amount of data in a relatively small amount
of time. We have tried to strike a balance between locally
stored data on the computer and the data retrieved from the
web while using the application. Information such as album art images, track names are not stored locally since they consume a huge amount of space, and are hence retrieved from web using the DataMiner while running the application. However the application is still functional using only the local database without any connection to the web. The current database takes around 1.5GB of space which can be easily shared or transferred in a smaller zipped format which is around 400MB.

The information retrieval module has been coded in C++ and uses the sqlite3 external libraries and API for connecting to the database and retrieving the data. There are mainly three types of retrieval functions:

1) Search - The main search functionality is invoked when the user searches for a term in the search field. All artists/labels which match the term exactly (limited to the first 10 exact matches) and which are most similar to the search term (limited to the first 40 matches for labels and artists each) are retrieved as shown in Figure 2.

2) Attribute queries - These include queries about an artists name, releases, aliases (different names of the same artist), bands, band members, releases, styles, genres, labels that the artist has released music under, and other information associated with an artist. For a release, these are queries about the release title, complete release information (masterID, styles, genres, year in which it was released, the labels it was released under). For a label, this information includes the releases released under that label, the styles and genres of those releases, the artists who released their music under that label etc. The information retrieved using these queries is displayed as in Figure 2(a).

3) Network link information queries - These queries form the basic framework for the network visualization. These serve to establish links like artist-artist, group - artist - group, artist - release - label, artist - release - artist, artist - track - artist, label - release - label, label - artists and so on.

4) Filtering query - We allow users to filter the releases based on various attributes like Genres, Styles, Formats, Item types etc. The state of selected variables is captured every time the filter button is pressed and a nested query is constructed to retrieve the filtered releases from the database. A view of the filters can be found in the right hand panel of the screens displayed in Figure 3 and 4.

C. DataMiner

This section of the code makes use of the libcurl library to access the Discogs web site while the user is exploring releases. Dataminer is invoked as soon as the user double-clicks on a release to display the information in more detail as shown in Figure 3(b). The release information obtained online is saved into the memory and passed to the parser to extract the track and credits information.

The Discogs API limits us to 5000 release requests daily so a lot of the code in DataMiner keeps track of how many release requests have been made to ensure that an error is returned once we pass the maximum daily limit. In addition to this we have optimized the code to save the release requests made to a cache so that if a user decides to return to a release he/she looked at earlier we can pull it up more efficiently from the file system.

D. Timeline Visualization

Timeline visualization as shown in Figure 4(a) allows the user to view the releases of artists or record labels in chronological order of release year or alphabetical order of release title. The visualisation is invoked after a successful search. The results are obtained by querying the database with attribute queries as explained in the Database Section C. A user can browse through the releases by scrolling the releases of a dot.
1) Views: There are two types of visual displays for time line visualisation, the album art timeline and Pin timeline. Each has its own advantage in terms of the amount of information presented in the screen and the amount of recognisable information. Album arts allow the user to readily recognise the releases, whereas Pins allow the user to view more releases at a time as can be seen in Figure 4(b).

2) Dot: After a successful search, the display contains releases grouped according to the view of the time line visualisation in a sorted order. Releases which are released in a given time span are stacked over a time dot. Releases whose titles start with a set of alphabets are displayed above an alpha dot. In short, alpha dot and time dots are the nodes which group the releases in Alpha-Numeric visualisation and Timeline visualisation. A user can choose the visualisation from the lower right panel of the application as can be seen in Figure 3 and 4.

3) Zoom-In/Zoom-Out: Controlling the step size is one of the important feature of the time line visualisation. The user can use the step size of the timeline to expand/reduce the span of years and increase/decrease the size of the alpha-numeric groups. Increasing or Decreasing the step size is equivalent to zoom-out/zoom-in feature respectively where, decreasing the step size allows the user to reduce the span of years contained by a time dot and gives a detailed view of the release for the reduced span and increasing the step size enables the user to assess the overall performance of the artist or label by expanding the span.

4) Bar-Chart: One of the interesting feature in the time line visualisation is to indicate the number of releases in a dot with respect to other dots. A normalized bar chart is displayed with each dot below the time line whose size is determined with respect to the number of releases released during the span of the dot. This quickly allows the user to assess the performance, success and contributions of an artist/label.

5) Scrolling-BarChart: Another interesting feature in the time line is to provide an indication of how many releases are yet to be discovered in a given dot. The current release position is highlighted in the bar chart with a black line. The scrolling of the releases is captured by a movement of a line in the bar chart allowing the user to track the amount of information left to discover and also account the information discovered.

6) Graphical Properties: Several graphical properties like circular effect, fish-eye view are included in the control panel as can be found in the lower right panel in Figure 3 and 4. The circular effect gives the timeline a 3D look as shown in Figure 4(a) and the fish-eye effects zooms into whatever the user is currently focusing on while scrolling the timeline. This slider allows the user to vary the fish-eyeness of the visualisation. Using the fish-eye can allow the user to view more releases in a screen.

7) Filtering: The user can filter the releases displayed by style, genre, members, formats, descriptions, and item type in the filtering window. The items in the lists are populated according to the results displayed in time line. Filtering is achieved by querying the data base with the Filter queries as explained in the Database Section. A view of the filter controls can be seen in the middle right hand panel in Figure 3 and 4.

8) Details-On-Demand: A user can find in-depth information, such as the description, the genre, the style, the year, format of the release, and the record label by double clicking the album that he/she finds interesting. The details-on-demand release window is displayed after a flipping animation of around 0.2 seconds. While, the animation happens the application invokes the DataMiner to request the release track information using discogs API. The requested information is parsed by the Parser module updating the structure required by the application to display the track and credits information. Figure 2(b) highlights this feature.

9) Switching to Other visualisations: An artist info overview window that shows the current artist with an image and a button allows the user to switch into the network visualization view. This can be found in the top right corner of the screen from Figure 4.

E. Network Visualization

The network visualization as shown in Fig 5(b) allows the user to view the connections of an artist. After a successful search of an artist, the network visualization will show a node representing the artist. A context menu as show in Figure 5(a) that allows the user to explore the connections appears by clicking on the node. The network can be explored in various ways: other aliases the artist goes by, releases that artist has, groups the artist is a part of, labels the artist has produced releases with, tracks the artist has contributed to, and compilations the artist has contributed to. In addition to this there is a button in the artist info overview window that allows the user to expand the connections graph to artists who share the same band as the artist we are currently viewing. The view control window displays the legends for artists, labels, releases, compilations and tracks as presented in the lower right panel of the Figure 5.

V. Evaluation

The effectiveness of MusicDigger was evaluated using a usability test of around five people. The form we came up with had three sections: A pre-evaluation questionnaire, a set of tasks to complete, and a post-evaluation questionnaire.

The pre-evaluation questionnaire was used to split users into three groups: music experts, music hobbyists, and casual listeners. Our application is targeted towards the music experts who make use of the discogs database but we realized that many other users could make interesting insights and enjoy the experience of discovering music as well. We also thought that these less experienced users could give us good insights into what works with our application and what does not.

The tasks of the user evaluation were laid out in such a way that each user was required to use all of the functionality within our application to complete all of the tasks. If any task was too difficult the user could skip it and he/she was encouraged to write about it in the post-evaluation questionnaire.
The post-evaluation questionnaire had three open-ended questions in which the user was asked to tell us what features of the application worked well, which features did not work so well, and what features they would like to be added in the next iteration of the application. Then there were a series of questions in which users were asked to rate the features of the application, the speed and responsiveness of the application, how aesthetically pleasing the application was, and how easy it was to view and understand the data being presented.

VI. FUTURE WORK/CONCLUSIONS

Our usability test gave us many interesting results. We found that most of the users did play instruments and so had some background in music. We found that the music experts were familiar with all the music web sites in our questionnaire (Discogs, Pandora, and Last.fm) and that the other groups, music hobbyists and casual listeners, were familiar with less.

Many of the users felt that the application was much faster and responsive than their typical experience searching for data on the web. All of the users also felt that the feature set was complete and had very little suggestions to make in this area. Users also seemed to enjoy the circular menu of the network visualization. We are also happy to note that all of the users were able to complete six of the seven tasks, it was only the final task that seemed to give them trouble. This leads us to believe that with a tutorial the application is very usable to answer insightful questions about releases in the Discogs database.

Some of the negative comments we received were that the information displayed was at times disorganized and overwhelming. This seemed to crop up the most in both the timeline visualization and network visualization with artists who were very well-known and had contributed a lot to the music world. Some users also mentioned that the Expand Artists With Shared Bands button could be moved into the circular menu instead of being placed in the artist overview window. Other complaints were that the album artwork was missing and that sorting releases by title was unintuitive for the timeline visualization.

There is a lot of room for future work. As it stands we are still making updates to complete all the features we had planned in the original iteration but that were unable to make it in time for the user evaluation. We also got lots of feedback from users who would like to see this as a commercial product. A lot of the discogs users who took our user evaluation expressed interest in paying a few dollars to use this product to browse the data on the web site. We also hope to help the music department integrate this product into some of their music courses.

VII. ACKNOWLEDGEMENTS

We would like to extend a sincere thanks to Robert Gibson, the Director of the University of Maryland School of Music, for being our faculty sponsor on this project and taking the time to meet with us this semester. His enthusiasm, support, and feedback helped us greatly improve the MusicDigger.
application. We hope that he will make use of it in some of his future music courses.

We would also like to thank Dr. Ben Shneiderman for providing us with valuable information as we studied the material presented in his information visualization course and for giving us great advice and feedback as we worked to complete this project. For the same reasons, we would like to thank Gleneesha Johnson, the teaching assistant for this course.

Lastly, but certainly not least, we would like to thank all who graciously donated their time to complete our user evaluation form and gave us the feedback we needed to make this project a success. With their acknowledgement, an incomplete list is as the following: Evren Ulusoy, Ahmet Gunes, Ferhan Ture, ...

REFERENCES


APPENDIX

A. Database Schema

Artist Table
1) int Artist ID (1 reserved for VARIOUS)
2) string Artist Name
3) string Image Url
4) int parentAliasID

Artist Member table
1) int groupID
2) int artistID

Label Info table
1) int Label ID
2) string Label Name
3) string ImageURL
4) int Parent ID

Release Main Table
1) int Release ID (assigned by DISCOGS)
2) string Release name
3) string Album Art url
4) int Master ID (not an actual release id)
5) int year

Release Genre Table
1) int Release ID
2) enum Genre ID

Release Styles Table
1) int Release ID
2) enum Style ID

MainArtist Release Table
1) int Release ID
2) int Artist ID

ExtraArtist ReleaseTrack Table
1) int Release ID
2) int Artist ID
3) string trackPosition
4) string/enum role

Compilations Table
1) int ReleaseID
2) int ArtistID
3) string trackPosition
4) string/enum role

Label Release table
1) int Release ID
2) int Label ID
3) string Catalog Number

Release Format table
1) int releaseID
2) int format
3) int quantity
4) int description