

Applications of Web Services to Digital Music Libraries

Mihai Parparita Brian Kernighan

Princeton University

May 5, 2003

Abstract

We present a system that attempts to impose an organized structure on a very large set of music files that lack any implicit arrangement. Issues such as a lack of metadata, incorrect information and human error are encountered in this process. We deal with them by using online services that maintain large databases of music information. Integrating data from these distinct services, as well as dealing with cases where even these services lack the information required, pose barriers as well. To resolve these remaining problems, algorithmic approaches are investigated.

1 Introduction

With the recent explosion of digital music libraries, enabling fast, accurate user access to their contents has become a significant task. Although simple lists of files and songs may suffice when working with collections whose sizes are measured in the hundreds of items, such methods begin to break down when thousands or even tens of thousands of songs are encountered. Such problems are further exacerbated when working across multiple collections, as distinct naming schemes, file arrangements, etc., come into play. The most common approach is to give up attempts at organization entirely and to instead resort to a query based interface that places the burden on the user, i.e. he or she must know at least something about the title, artist or album of a song for it to be found. Additionally, such systems usually do not provide a "context" for a song, meaning what when it is found, it is difficult to locate other songs that are on the same album, recorded by the same artist or in the same musical genre.

We present a system entitled "pTunes" that attempts to solve all of these problems by using web services in addition to more traditional data mining approaches. To demonstrate its advantages, we apply it to the music found on the Princeton University LAN in the month of April 2003, totaling 86,072 files. In addition to providing us with a large dataset, these files originate from 209 distinct users, thus also allowing us to tackle the related problem of integrating multiple music libraries. We believe that the end result is a system that enables intuitive browsing of a large quantity of music.

2 Previous and Related Work

Because of the popularity of digital music libraries, "jukebox"-like systems that help users manage their collections are prevalent. Programs such as iTunes [6], WinAmp [9] and MP3 Voodoo [7] all provide some measure of organization. Of these, WinAmp is representative of the most primitive organization scheme, since all it provides is a long scrolling list of songs. iTunes builds on this by using a three-paned interface to represent artists, albums and songs. Finally, MP3 Voodoo allows multiple "views" of a collection, for example by year, artist or album cover. However, all of these systems depend on user input and initiative, e.g., in the last example the user must provide all album cover art. This may be feasible if the user is building up his or her collection gradually and adds this information progressively. However, when importing one's existing, possibly large, music collection into a catalog system, such expectations are unreasonable.

Work in this area has been done from the academic perspective as well. For example, [2]

attempts to use automatic genre classification to organize collections. However, the approach presented suffers from several drawbacks. First, an exclusive reliance on automatically extracted information means that subtle variations in categorizations cannot be captured, and accuracy is bound to be lower. Secondly, the mode of presentation is a 2D map where songs in the same genre tend to cluster together. Although this may be reasonable with small-sized music collections (e.g., under 1,000 files), such a system does not scale (due to visual clutter) when dealing with something like our dataset.

3 Data Normalization

The choice of MP3 files as a dataset had the advantage that this format has a reasonably well developed system for metadata. In addition to audio data, files can also contain ID3 tags, which contain information such as artist, album, title and genre. The first three were used to index files in a large database. This has the added advantage that duplicate files would be referenced under a single song, which helped to reduce visual clutter for the user.

However, in order to efficiently group files by song, the restrictions of matching had to be reduced. Variations caused by such things as typos, misspellings and abbreviations meant that what to the human eye represents the same song would end up with two distinct entries in the database. A two pronged approach was chosen to resolve this problem. Firstly, all names were "normalized," that is made lowercase, stripped of white space, with numbers expanded to their word equivalents, common prefixes and suffixes such as "the" and "an" removed, etc. However, to preserve the familiarity of the original names to users, all of these operations were done on secondary copies of the strings that pTunes used internally to keep track of files, while the

human-readable versions were presented in the interface.

Though this covered some of the variations, the issue of misspellings still remained. To resolve this problem, a Perl module, `String::Similarity`, which implements a fuzzy string comparison as described in [1], was used. With over 50,000 distinct title, album and artist strings, a direct N x N comparison to find similar strings would have been unreasonable. Therefore, the assumption that at least the first letter of a name would be correct was made, thus significantly reducing the amount of searching. In this way, similar strings could be grouped together, and a "canonical" one could be elected to replace all others in its group. The one with the most appearances was chosen for this, though correctness was not an issue, since as previously stated normalized strings are not visible to the end-user.

Although a large degree of similarity was required for two strings to be declared equivalent, there could still be false positives. That is, the system has no way of knowing (for example) that "Breeder" and "Breeders" are two distinct groups. As a workaround for this, a small companion web application was created that could be used to flag false equivalences. An administrator of such a system is presented with a list of possible substitutions, and seemingly incorrect ones are can be flagged such that they are not taken into account (in the present pass as well as future ones).

stylesredmanandmethodman	redmanandmethodman	<input type="checkbox"/>
suckers	smuckers	<input checked="" type="checkbox"/>
sunnycamehome	sonnycamehome	<input type="checkbox"/>
sunnysideofstreet	onsunnysideofstreet	<input type="checkbox"/>
sunstorm(originalmix)	sandstorm(originalmix)	<input type="checkbox"/>
swell	sowell	<input type="checkbox"/>
swingoriginalbroadwayrecording	originalbroadwayrecordings	<input type="checkbox"/>
swingwhenyourewinning	singwhenyourewinning	<input type="checkbox"/>
system7	systemf	<input checked="" type="checkbox"/>
tannenbaum	otannenbaum	<input type="checkbox"/>
thavenueheartache	sixthavenueheartache	<input type="checkbox"/>
theirlaw	theiria	<input type="checkbox"/>
theme	themes	<input type="checkbox"/>
themefromschindlerslist(r	themefromschindlerslist	<input type="checkbox"/>
thepointersis	pointersisters	<input type="checkbox"/>

Figure 1: False Substitution Flagging

There was also the issue that some files (about a quarter of those encountered) did not possess some or all of the artist/album/title metadata that interests us. In such cases an alternative path was taken, with the file names and directory structures (if any) being used to extract information. This approach was based on the observation that a few naming schemes such as "<artist> - <title>.mp3" or "<track number>. <artist> - <album> - <title>.mp3" were common. These naming schemes usually originate with the software that was initially used to encode the MP3 files, though occasionally more dedicated users will attempt to re-label things with such a consistent naming scheme themselves. For the purposes of this system, files for which no discernible information could be extracted at all were ignored.

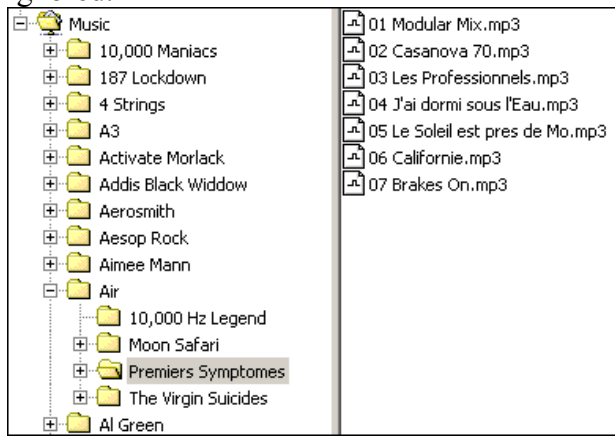


Figure 2: Typical user file organization

4 Genre Estimation

We felt that the most crucial piece of information needed to enable browsing of songs is genre. Users are most likely familiar with the organization of traditional record stores, which usually separate artists by the type of music they generally create. Our system is somewhat more flexible, i.e. we can have more specific genres since we have no shelf-space to worry about, and we can also allow an artist to be listed under multiple genres. For example, Antonio Carlos Jobim's

songs, such as "Girl From Ipanema" can be described as falling under Jazz (at the most generic level) and "Latin Jazz" more specifically, but also at the same time under "World Latin Music" and "Bossanova." Being able to determine all these genres accurately meant that no matter which genre the user felt it feasible for the artist should fall under, it would be listed there. Therefore, choosing genres for songs then became the main purpose of our system, as well as the main metric by which we could evaluate its success.

4.1 Internal Metadata

As previously mentioned, genre is one of the pieces of metadata that can be contained within MP3 files. When indexing files, extracting this along with the other necessary data proved to be very easy. Since this genre information is present per song, and we are concerned with categorizing artists, this meant that we could get multiple pieces of data per entry. A rough system of weights was devised, where each genre had a number attached that was proportional to the number of songs that used it.

4.2 FreeDB Integration

FreeDB is a large online database that attempts to collect metadata relevant to CDs in circulation (specifically including artist, album title, track titles and genre). Its main purpose is to serve as an easy way to obtain track labels for compact discs played in computers. As a side effect of this aim, it has accumulated a very large dataset (currently 947,185 CDs) that attempts to encompass all albums ever released. This data is all user submitted; the common usage pattern is for a user to insert a CD into the computer, which then looks up the album (specifically a hash based on track lengths) in the database. In the (common) case that the information is already there, the data is simply copied locally. If it is not, the user is

given the option of inputting it on the spot and submitting it back to the server, so that others may benefit in the future.

Due to the liberal policies of its maintainers (FreeDB was started in response to CDDDB's (an earlier service) decision to claim all album metadata as being their property), it is possible to obtain the entire dataset that FreeDB has accumulated and use it at will. The pTunes system relies in part on a local copy of the database that has been imported into a SQL table and indexed by normalized artist (the original database is in a directory/file structure that is indexed by CD hash, which is not ideal for our purposes).

To use this data within pTunes, the desired artist is looked up in the table, and the genre information for all albums of that artist is extracted. Chances of matches are high; most of music files have as their source encoded CDs, with said CDs usually obtaining their track information from FreeDB. Genre information is not always present (since it is not the main purpose of the database), but given an artist with at least a couple of albums, chances are that at least one will contain this information.

Although FreeDB only provides one genre datapoint per album, this data is viewed as being more significant than the information contained within files. Therefore the weighing applied to this is directly proportional to the number of tracks that were present in the album (the idea being that, if each track was represented by a file that had the correct genre information, each one of them would contribute an entry in the genre table with that string).

4.3 Amazon.com Integration

Amazon.com, a large online retailer, is also in possession of a large database of music information, due to the number of CDs that it lists in its catalog. Amazon exposes this data using their "Amazon Web Services" interface

that supports an URL based query scheme that returns XML documents as well as a more formalized SOAP-based method. Because Amazon is a commercial entity, which in part derives its success from the quality of its catalog, it was hoped that data obtained from this service would be of superior quality to other sources. Additionally, Amazon is a source of other information about albums, specifically cover art and related CDs, that is appropriate to have in a system that aims to encourage browsing.

To interface with Amazon, the URL query-string based interface is used, since its simplicity seems more appropriate to the very rigid and structured way in which the database would be queried: a simple search by artist name and album. The SOAP interface seemed more appropriate for situations in which more flexible queries were necessary. The service offers the option of applying a user-specified XSLT (XML Stylesheet Transform) to the returned XML document, thus allowing data to be massaged to a form that is easily parseable.

One barrier was encountered: although the XML documents should in theory contain the category that the product is listed under (in the case of music, this is most often a genre), almost none of the returned results seemed to actually have these fields. Comparisons with the regular, HTML-based store interface revealed that it was present there, thus further confounding the issue. In the end, the "clean" approach of the XML interface had to be "dirtied" a bit, since a secondary query that parsed the HTML version of the product page had to be done in order to obtain genre information. There was still an advantage to using the XML interface first; it could be used for album cover image URLs, recommendations and most importantly ASINs (Amazon product IDs). These ASINs could then be used to immediately select the proper HTML page to parse for genre information.

4.4 Sound Analysis

Though the above data sources can be used when enough information is available such that a song can be looked up, there are cases where this is not possible. Therefore, as a last resort the pTunes system attempts to perform automatic genre classification. This is done through the MARSYAS system described by [3]. Briefly, this system attempts to extract features from audio data (e.g., timbre, texture and instrumentation) and then to match them against one of ten genre templates that it has been trained for. The system claims 60 to 70% accuracy, comparable to the abilities of humans to discern musical genre. The fact that this system could be plugged into pTunes with very little modification (the analysis component is a command-line utility) made it very appealing.

As a general mode of operation, this pass is done after all other attempts at categorization have failed. Remaining artists are listed, and a few songs for each one are chosen and copied locally. After some post processing (MARSYAS expects input to be single-channel sound sampled at 22 KHz) each file is fed in turn to the analyzer and the results are recorded in the database. Due to performance considerations (analysis ran at 2x real-time on the development system), only 30 second samples of each songs were examine; however past experiences with the MARSYAS have shown that analyzing the full song provides only marginal improvements.

4.5 Genre Data Uniformity

Though the various genre extraction methods each provided their results, the issue of integrating all of them into a coherent whole still remained. Genres stored in the files amount to 461 distinct strings, while the data extracted from FreeDB contributes a further 5,860 and Amazon adds 1,459 more (the 10 that MARSYAS adds are insignificant by comparison, and are a subset of the strings

contained in the files). Using all of these as is is not an option; browsing through a list of 7,780 genres is no better than going through the entire list of 9,994 artists. Therefore the decision was taken to collapse all of these genres into 140 "canonical" ones (listed at the end of this report). There were two aspects to this grouping. One was that the exact same genre was represented in more than one way (e.g. "Alternative Rock" can also be written as "Alt. Rock") and the other was that some labels were too specific (e.g. "Modern Romanian Pop", of which only one song was present, was better listed under "Eastern European" music). Thus while the first grouping was of a very similar nature to the string substitution that had to be done to normalize artist/album/strings, the second group is of a more semantic nature, with some awareness of musical genres and hierarchies being required.

The canonical genres were chosen somewhat informally, with Amazon's system giving the rough framework, which was then altered where personal experience suggested something more appropriate. To create the "mapping table" for collapsing, a somewhat brute force approach was taken. Using a small helper application, a few hundred genre strings could be viewed at one time and each could then be mapped to the canonical genre that it would fall under. Given enough time and perseverance, this resulted in most of the input genre strings being assigned to the right canonical one.

Using these genre mappings and the information extracted from the internal tag, FreeDB, Amazon and MARSYAS data sources, it was now possible to attempt to categorize each artist into one or more genres. The approach taken was to obtain a list of all distinct canonical genres that were reported for the artist, and to then assign a weight/frequency count for each genre (as described for each preceding data source). Then the average frequency was computed, and any genres which had a weight at least a third

of the value of this average were included (the 1/3 value was chosen empirically). This prevented bad categorization information from skewing results too much, while still allowing eclectic artists to be listed in multiple categories.

5 User Interface

To present all of the acquired information, one of the previously mentioned interface paradigms was extended. The iTunes 3-pane interface underwent a few extensions to accommodate genre information, album covers, recommendations and the general sense of "context" that we wanted to provide. First, an additional pane was inserted such that the hierarchy was now genre -> artist -> album -> song, as shown in Figure 4. A more traditional query based interface was still retained, as evidenced by the search box in the upper right corner. Search results would then appear in the songs pane, and could be operated on normally, just as if the user had browsed to them.

Then, an Info Bar was created that would show information relevant to the current song. This included the cover of its album (as extracted from Amazon), the genres that were extracted for that artist, similar albums as recommended by Amazon, etc. An example of the Info Bar as applied to a specific album is shown in Figure 5.

This entire interface was hosted within a web browser, partially to ensure cross-platform compatibility, but also to leverage some web concepts. Specifically, the concept of linking was heavily used to show relationships between items. For example, in the Info Bar described above the listed genres are actually links that lead to lists of other artists in that same category. Also, when displaying songs in the song pane, artist and album names are links as well, so that the user can easily see the entire contents of an album or the discography of an artist. With further work on the backend, this linking concept could be expanded such that

remixes of one song, or different performances of a single classical piece could all be seen at a glance.

6 Results

As previously mentioned, genre categorization was used as the metric by which the organization abilities of our system were measured. As a baseline, we looked at the number of artists that could not be placed in a genre using only the information contained within the files themselves (i.e. the MP3 ID3 genre tag), which turned out to be 43%. Further algorithmic improvements (for example, when encountering the artist string "<artist A> with <artist B>", the data present for "<artist A>" and "<artist B>" by themselves was used as well) brought this number down to 38%.

Leveraging the FreeDB database lowered the proportion of uncategorized artists from 38% to 23%, a 40% improvement. For a database that claims to contain every single CD in existence, this is perhaps somewhat disappointing. Several reasons can be postulated to account for this. One is that FreeDB's ASCII-centric data makes searching for international titles difficult. For example, the album and song titles of Japanese artists UA are listed in their Romanized equivalents, while a user is much more likely to label things using the original Japanese characters. Furthermore, since genre is not its primary concern, this information was sometimes missing from album entries. Finally, since FreeDB is fundamentally a user submitted service, it is also sensitive to typos, not all of which may be compensated for with the substitution engine.

When the dataset obtained from Amazon was integrated, the uncategorized percentage now stood at 20%. This 13% improvement over 23% is perhaps even more disappointing than the one obtained from FreeDB. One reason for this is that there is most likely a great degree of overlap between the two

datasets, since they both focus on commercially available CDs. However, since Amazon's data is provided by paid employees, it tends to be more consistent and less prone to error.

Finally, when genre analysis was applied to the remaining artists, the number of uncategorized ones theoretically dropped to zero. However, since the MARSYAS system only makes loose guarantees about accuracy, chances are that some of the artists are mislabeled.

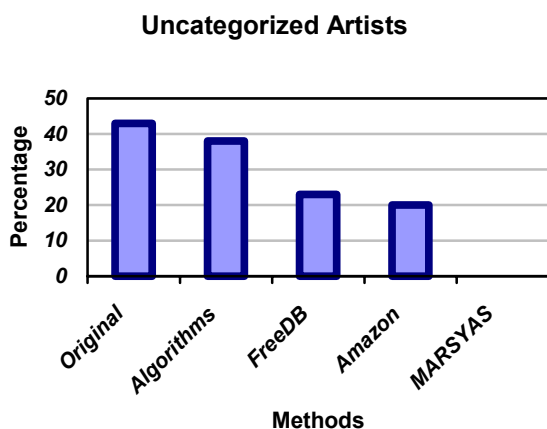


Figure 3: Genre Assignment Results

7 Discussion

As a whole, the system mostly achieved its task. That is, it is possible to have a coherent browsing experience across a large dataset that originates from multiple people. Informal tests with users revealed that most were able to find what they were looking for, and in situations where they could not, it generally was the case that the song was not present at all. Users were generally pleasantly surprised by the organization structure, but that is mostly because their exposure to large scale music libraries has been in the form of peer to peer applications such as Kazaa. Those systems make very few attempts at categorizing or even cleaning up existing data using any sort of "normalization."

Since the pTunes system was developed rather organically, several concepts applied later on in the project were not used in other portions. For example, since the FreeDB database contains a large list of artists, it could be used to flag most of the false positives that the substitution engine runs into. For example, a simple search would reveal that both "Breeder" and "Breeders" are artists in their own right, and thus one should not be mapped to the other. Similarly, the manual canonical genre mapping process could be accelerated by applying some of the techniques developed to deal with typos, abbreviations and misspellings in artist/title/album normalization. Though this would not help with the semantic mappings previously mentioned, it would remove most of the mechanical aspects of the genre mapping process.

8 Future Work

As previously mentioned, though the system has succeeded in categorizing all of the music that is available, correctness is another issue entirely. Two approaches that could alleviate this come to mind. Firstly, users could be induced to validate the data themselves. In a web environment with multiple users, it is likely that some will care enough about their favorite artists to make sure that they end up in the right genre. Given a system that is easy enough to use (e.g. a "Disagree?" link next to the genres listed in the Info Bar that leads to a simple form where a new genre can be chosen), this could be a feasible approach. There is still the issue that users may themselves provide inaccurate data, but a majority rules or moderation system should counteract this. Similar systems have already been deployed on sites such as Slashdot [8], which must cope with a large quantity of user comments of varying quality.

Another approach to consider is to leverage another web service, specifically the Google search engine. If an artist is categorized

under a genre, chances are that there exist web pages mentioning both the artist and the name of that genre. Using Google's web services API, it should be possible to perform some basic verification. Google can also be used to enhance the concept of misspelling and typo detection, since it also uses a "majority rules" heuristic to correct search terms, except that its dataset is several orders of magnitude larger than the one pTunes uses, and thus could be much more effective.

Although the current system has recommendations obtained from Amazon, it should be possible to extend it such that the dataset itself is used to set up correlations between artists, albums and songs. Specifically, since the data is originally divided up into collections, and persons A and B both have artists I and II, then someone looking at artist I may be interested in II and vice versa. Such approaches are already being investigated by [4].

9 References

- [1] Eugene Myers, "An $O(ND)$ Difference Algorithm and its Variations." *Algorithmica* Vol. 1 No. 2, 1986, pp. 251-266;
- [2] Andreas Rauber and Alexander Müller-Kögler, "Integrating Automatic Genre Analysis into Digital Libraries." *ACM/IEEE Joint Conference on Digital Libraries*
- [3] George Tzanetakis and Perry Cook "Musical Genre Classification of Audio Signals." *IEEE Transactions on Speech and Audio Processing*, 10(5), July 2002
- [4] <http://echocloud.net/>
- [5] <http://search.cpan.org/author/MLEHMAN/N/String-Similarity-0.02/>
- [6] <http://www.apple.com/iTunes/>
- [7] <http://www.mp3voodoo.de/>
- [8] <http://www.slashdot.org/>
- [9] <http://www.winamp.com/>

Canonical Genres

Blues

- Chicago Blues
- Electric Blues
- Country Blues
- Delta Blues
- Texas Blues
- Female Vocal Blues
- Zydeco Blues

Classical

- Avant Classical
- Chamber Music
- Classical Guitar
- Composers
- Opera
- Solo Instrumental
- Symphony

Country

- Alt Country
- Bluegrass
- Contemporary Country
- Country Rock
- Traditional Country
- Honky Tonk
- Rockabilly Revival
- Western Swing

Electronic

- Acid Jazz
- Ambient
 - Abstract Ambient
 - Dark Ambient
 - Soundscape
- Breaks
 - Big Beat
 - Breakbeat
 - Funky Breaks
 - Nu Skool Beats
- Dance
 - Live & DJ
 - Club
- Downbeat
 - Dub Techno
 - Lounge
 - Trip Hop
- Drum 'n' Bass
 - Ambient Drum 'n' Bass
 - Breakcore
 - Darkstep
 - Drill 'n' Bass
 - Intelligent Dance Music
 - Jungle
 - Techstep
- House
 - Ambient House
 - Deep House
 - Disco House
 - Garage
 - Speed Garage
 - Happy Hardcore
 - Hard House
 - Organic House
- Industrial
 - Coldwave
 - Electronic Body Music

Techno

- Acid Techno
- Detroit Techno
- Electro Techno
- Gabber
- Happy Hardcore
- Rave
- Tech House
- Trance
 - Ambient Trance
 - Goa Trance
 - Hard Trance
 - Melodic Trance
 - Progressive Trance
 - Psytrance

Experimental

- Electroacoustic
- Environments
- Improvisation
- Minimalistic
- Noise

Folk

- 60s Revival
- Anti-Folk
- British Folk
- Contemporary Folk
- Singer-Songwriter
- Traditional Folk

Heavy Metal

- Funk Metal
- Hair Metal
- Industrial Metal
 - Grindcore
- Nu Metal
- Rap Core
- Trash
 - Death Metal
 - Black Metal
 - Doom Metal
 - Speed Metal

Hip Hop

- Abstract Hip Hop
- Bass
- Gangsta Rap
- Independent Hip Hop
- Turntablist
- Old School Hip Hop
- Pop Rap
- Southern Hip Hop

Jazz


- Be Bop
- Cool Jazz
- Hard Bop
 - Avant Garde
 - Post Bop
- Latin Jazz
- Soul Jazz
 - Jazz Fusion
- Big Band
 - Dixieland
 - Swing
- Crossover Jazz
- Lounge

- Vocal Jazz
- Modern Rock**
 - Alternative Rock
 - Adult Alternative
 - Brit Pop
 - Grunge
 - Spanish Rock
- Experimental Rock
- Indie Rock
 - Chamber Pop
 - Indie Folk
 - Indie Garage
 - Indie Pop
 - Jangle Pop
 - Lo Fi
 - New Psychedelia
 - Noise Rock
 - Post Rock
 - Space Rock
- Jam Rock
- New Wave
 - Goth Rock
 - Synth Pop
- Post Punk
- Power Pop
- Oldies**
 - Doo Wop
 - Early Rock & Roll
 - Rockabilly
 - Surf
- Other**
 - Acapella
 - Children's
 - Comedy/Spoken Word
 - Contemporary Christian
 - Film Scores
 - Novelty
 - Seasonal
 - Show Tunes
- Pop**
 - Dance Pop
 - Easy Listening
 - Euro Pop
 - J-Pop
 - New Age
 - Soft Rock
 - Teen Pop
 - Vocalists
- Punk Rock**
 - '77 Style Punk
 - Cow Punk
 - Hardcore Punk
 - Emo
 - NY Hardcore
 - Oi!
 - Pop Punk
 - Proto-punk
 - Psychobilly
 - Riot Grrrl

- Ska Punk
- R&B**
 - Funk
 - Disco
 - G-Funk
 - Gospel
 - Soul
 - 70s Soul
 - Contemporary R&B
 - Motown
 - New Soul
 - Quiet Storm
- Reggae**
 - Roots Reggae
 - Dub Reggae
 - Dancehall
 - Lovers Rock
 - Pop Reggae
- Ska
 - Rock Steady
- Rock**
 - Classic Rock
 - Acid Rock
 - Garage Rock
 - Blues Rock
 - British Blues Rock
 - British Invasion
 - Folk Rock
 - Glam Rock
 - Prog Rock
 - Kraut Rock
 - Southern Rock
 - Hard Rock
 - Guitar Rock
- World**
 - African
 - Traditional African
 - Afro Pop
 - Afro Beat
 - Mbalax
 - Rai
 - Asian
 - Indian Classical
 - Bombay Pop
 - Qawwali
 - Celtic
 - Eastern European
 - Klezmer
 - Indigenous Music
 - Latin
 - Calypso
 - Merengue
 - Son
 - Salsa
 - Samba
 - Bossa Nova
 - Tropicalia
 - MPB
 - Western European



Figure 4: pTunes overall interface

Song
I'll Go Dreaming
Album

Artist
<i>BT</i> is listed under the following genres:
<ul style="list-style-type: none"> ■ Electronic ■ Electronic : Dance ■ Electronic : Trance ■ Styles : Dance & DJ : House ■ Styles : Dance & DJ : Trance ■ Styles : Dance & DJ : General
Recommendations
<ul style="list-style-type: none"> ■ <i>Escom</i> by <i>BT</i> ■ <i>Tranceport</i> by <i>Paul Oakenfold</i>

Song
(Agnelli + Nelson) - El Nino
Album

Artist
<i>Paul Oakenfold</i> is listed under the following genres:
<ul style="list-style-type: none"> ■ Electronic ■ Electronic : Dance ■ Electronic : House ■ Electronic : Techno ■ Electronic : Trance ■ Indie Music : Dance & DJ : General ■ Other ■ Styles : Dance & DJ : General ■ Styles : Dance & DJ : Trance ■ Styles : Soundtracks : General ■ Other : Film Scores
Recommendations
<ul style="list-style-type: none"> ■ <i>Voyage in to Trance</i> by <i>Paul Oakenfold</i> ■ <i>Out There And Back</i> by <i>Paul Van Dyk</i> ■ <i>Bunkka</i> by <i>Paul Oakenfold</i>

Figure 5: pTunes Info Bar detail and linking example