Mining Music Social Networks for Automating Social Music Services*

Claudio Baccigalupo and Enric Plaza

IIIA - Artificial Intelligence Research Institute
CSIC - Spanish Council for Scientific Research
Campus UAB, 08193 Bellaterra, Catalonia (Spain)
Vox: +34-93-5809570, Fax: +34-93-5809661
Email: {claudio,enric}@iiia.csic.es

Abstract. Community-driven services compile data provided by the community members, for instance playlists in Web 2.0 music sites. We show how this data can be analysed and knowledge about sequential associations between songs and artists can be discovered. While most of this kind of analysis focus on (symmetric) similarity measures, we intend to discover which songs can "musically follow" others, focusing on the sequential nature of this data in a database of over 500,000 playlists. We obtain a song association model and an artists association model, we evaluate these models comparing the results with other similarity-based analysis, and finally we show how these models can be used to automatically schedule sequences of songs in a social Web radio service.

1 Introduction

In our view, the most interesting data in community-driven services (also called social network Web sites or Web 2.0 applications) are those provided by the community members themselves, the reason being this data would simply be unavailable (or inexistent) if they were not provided by the components of such communities. An example is found in music-related Web communities that allow people to share their personal playlists, which may have been compiled for different purposes (e.g., to listen while jogging, working, partying).

The success of many Web-based communities is also related to the creation of services which target the *social nature* of their public, that is, services that exploit knowledge about people in relation to other people, rather than about individuals. An example of such a service, described in this paper, is that of *Web social radios*. In a Web social radio, a group of people listens at once to the same stream of music, on the same Web radio channel. This service is interestingly different from the so-called Web *personalised* radios, that generate music streams individually tailored for every single person, who will listen in isolation.

* This research is supported in part by a MyStrands scholarship and by MID-CBR project TIN2006-15140-C03-01.

Poolcasting is a Web social radio we have developed, that *customises* the music of each channel for its *group* of listeners, combining a Case-Based Reasoning approach [3] with a knowledge discovery process over a Web 2.0 community. In this paper we focus on the knowledge discovery process needed to construct a model of "musical association" among songs and among artists. The knowledge discovery process has the goal of determining meaningful associations of songs and artists to insure that the sequence of songs streamed into a radio channel is not only customised for the audience group, but also contains a "meaningful" musical sequence, as if it were compiled by a human DJ. This knowledge discovery process has been applied to data the Web 2.0 music community MyStrands has collected from its users. In particular, we have accessed the dataset of playlists the members of MyStrands have shared on the community Web page. By analysing this database of playlists, we were able to find out, for each song, which other songs are more suitable to follow in a good sequence of music.

Several researchers assume that playlist merely contain "similar music"[8]. However, our assumption is not that songs in a playlist are "similar", but that when songs co-occur sequentially in one playlist, they musically flow one after the other, that is they "sound well together" *one after the other* in some specific context (e.g., jogging, working, partying). Although we ignore the implicit "meaning" of a given playlist, we assume that the sequence in which the songs are ordered "makes sense", so that if two songs are contiguously one after the other in a playlist, it somehow "makes sense" to play them in this order (the same assumption is made in [1]). Although this assumption may not hold for *each* playlist, applying this hypothesis to a large dataset of playlists implies that we can extract information about songs that are associated when they co-occur together in many playlists; in particular when they co-occur contiguously and in the same order.

2 User provided playlist data

There are different online sources for playlists. The most relevant ones are Web-based communities dedicated to share playlists: The Art of the Mix, Fiql, GoFish, UpTo11, WebJay, and MyStrands[1]. We work with MyStrands since we have direct access to the user playlists, although these can also be accessed using the Web API OpenStrands, available for developers.

Dynamic nature of playlists. A common thing for Web 2.0 applications is the dynamic nature of user-created content, that can be continuously updated (consider for example *wikis* and *blogs*). Indeed, MyStrands users can publish their personal playlists on the community Web page and later access them to add/remove songs or entire playlists. This fact increases the quality of the data, for new songs are easily entered into playlists, without any hard technical obstacle for users. The larger the number of available playlists, the best the quality of the discovered knowledge.

[1] Their Web pages are respectively: http://artofthemix.org, http://fiql.com, http://gofish.com, http://upto11.net, http://webjay.org, and http://mystrands.com.

Contentless nature of playlists. A playlist on the Web contains only a sequence of "referrers" to some songs, not the actual songs. Unfortunately, there is no such a thing as an universal ID that allows to univocally represent a song in a playlist. Moreover, new songs appear everyday: hence, the problem of correctly identifying *which* songs are included in a playlist. Since we work with MyStrands, we solve this problem by using MyStrands identifiers that are automatically assigned to every song in their catalogue (6 millions and growing).

Social nature of playlists. Playlists are published by the members of the MyStrands community in two different ways: either via the Web page, by manually adding a track after the other to a *current* playlist, or via the MyStrands media player plug-in, that allows to publish playlists directly from the media player (iTunes or Windows Media Player). The dataset of playlists that we have used is a "static snapshot" of all of these user playlists, taken on March 7, 2006, and containing 599,565 playlists.

There are some properties about the MyStrands members, songs and playlists that are worth mentioning: (1) members are 65% male, 35% female, are 32 years old in average (standard deviation: 10 years), and come from a number of countries (United States 41%, Spain 23%, United Kingdom 5%, Canada 3%, Germany 3%, others 25%); (2) the genres of the songs are unevenly distributes since Rock has 58%, and the rest are R&B 7%, Electronic 6%, Latin 5%, Soundtracks 5%, Jazz 3%, Rap 2%, others 14%, and as a consequence we expect to find better results for the most frequent genres; (3) the average length of a playlist is 16.8 songs (standard deviation: 11 songs); very infrequent songs can be found in playlists as well as very popular songs, and this will be reflected in our results.

Noisy nature of playlists. To analyse different playlists from different users and/or different datasets, we need some uniformity mainly about two things: a) the format of a playlist b) the way to identify a track (or artist, or album). Concerning the first point, there exists a quite-standardised format of storing a playlist, via the XML Shareable Playlist Format (XSPF), and more and more communities (including MyStrands) are moving towards this format. Concerning the second point, this is an open issue; every community uses its own IDs to refer to a track, and the same track can sometimes be referred to with different IDs (e.g., a studio version and a live version of the same song). In our case, we basically skipped this problem of track identification by using the IDs that were provided by MyStrands. From their IDs, we had to exclude some indexes that were referring to *virtual* elements; for instance the ID that corresponds to "Various Artists", which cannot be considered as the *same* artist every time it occurs.

3 Discovering Associations by Usage

In this section we will analyse playlist data to discover *song association degrees* and later *artist association degrees*.

Let $s(X, Y) \in [0, 1]$ be the *song association degree* from a song X to a song Y. Counting just the frequency with which two songs appear together

in a collection of playlists is not sufficient to estimate their association degree, for some songs are quite rare, but still are strongly associated with other rare songs. One solution is to consider the association strength from song X to song Y as the conditional probability to find song Y, given a playlist that contains song X, i.e., $P(Y|X) = \frac{f(X,Y)}{f(X)}$, where $f(X)$ is the *popularity* of X (defined as the number of playlists where X appears). Notice that $P(X|Y) \neq P(Y|X)$: the relation is not symmetric. This measure is biased towards having high conditional probabilities with songs that are very popular. That is, $P(Y|X)$ may be high as a result of the fact that Y occurs very frequently and not because X and Y are strongly associated. We correct this problem dividing $P(Y|X)$ by a quantity that depends on the popularity of Y: if Y is very popular (say, more than the average), the association degree is decreased, otherwise it is increased; the exact degree of scaling depends on the playlists and on the distribution of popularity among songs. The following formula takes into account these factors to compute the association between two songs X and Y:

$$\frac{f(X,Y)}{f(X) \cdot (f(Y)/\overline{f})^{\beta}} \tag{1}$$

where $\overline{f}$ is the average song popularity, and β is a parameter that takes a value in $[0,1]$; when $\beta = 0$, the function is identical to $P(Y|X)$.

We improve this measure by taking into account how far apart two songs are in a playlist, and their relative order. This can be done using a "sliding window" that lists a certain number of consecutive songs in a playlist: if two songs co-occur inside this window, they are considered to be associated, otherwise not. In this way, songs that are common but not specifically associated will not co-occur often relatively to the total number of their occurrences.

We make three assumptions: 1) the farther two songs occur in a playlist, the smaller is their association; 2) if two songs are separated by more than a threshold of $\delta \geqslant 1$ songs in a playlist, their association is null; 3) any song X is more associated to the songs it follows in a playlist than to the songs it precedes. This last point can be explained as follows: since our final goal is to program a radio channel by selecting one song *after* the other, and since the order between songs can be meaningful (e.g., the end of a track mixes into the beginning of the next one), we endeavour to preserve it.

Let $\mathcal{Q}$ be a collection of playlists and $q \in \mathcal{Q}$ be one of these playlists, $q = (X_1, X_2, \dots)$. Let X and Y be two songs; we denote as $d(q, X, Y)$ the distance that separates them in q, e.g., $d(q, X_i, X_j) = j - i$. If either X or Y does not occur in q, $d(q, X, Y) = \infty$. The songs X and Y are associated in q if $d(q, X, Y) \leqslant \delta$; formally we define their *song association degree in q* as:

$$w(q,X,Y) = \begin{cases} 0 & \text{if } |d(q,X,Y)| > \delta \\ 1/|d(q,X,Y)| & \text{if } |d(q,X,Y)| \leqslant \delta \wedge d(q,X,Y) > 0 \\ \alpha/|d(q,X,Y)| & \text{if } |d(q,X,Y)| \leqslant \delta \wedge d(q,X,Y) < 0 \end{cases}$$

where $\alpha \in [0,1]$ is a parameter to assign higher associations to post-occurrences than to pre-occurrences. Finally, to estimate the *song association degree* between

X and Y, we substitute in Eq. 1 the numerator with $\sum_{q \in \mathcal{Q}} w(p, X, Y)$. That is, rather than accumulating 1 for each playlist q where X and Y co-occur, we accumulate $w(q, X, Y)$, which equals 1 only if Y occurs contiguously after X in q, otherwise $0 \leqslant w(q, X, Y) < 1$:

$$s(X, Y) = \frac{\sum_{q \in \mathcal{Q}} w(q, X, Y)}{f(X)(f(Y)/\overline{f})^{\beta}} \quad . \tag{2}$$

From the MyStrands dataset of playlists, we obtain an average song popularity $\overline{f}$ of 37. We set parameters $\alpha = 0.75$, $\beta = 0.5$, $\delta = 3$ and discard any song that occurs just once, as well as associations within the same artist, for their obviousness. The result is a set of 112,238 distinct songs that have a non-null association with some other song. For instance, the top associated tracks found for *Smoke On The Water* (Deep Purple) are: *Space Truckin'* (VV.AA.), *Cold Metal* (Iggy Pop), *Iron Man* (Black Sabbath), *China Grove* (The Doobie Brothers), *Crossroads* (Eric Clapton), *Sunshine Of Your Love* (Cream), *Wild Thing* (Jimi Hendrix).

We have analysed the same collection of MyStrands playlists to discover knowledge about artist association. Given a playlist $q = (X_1, X_2, \dots)$ and two artists A and B, we denote as $a(X_i)$ the artist of the song X_i, and we denote as $d'(q, A, B)$ the minimum distance that separates a song of A and a song of B in q, e.g., if $a(X_i) = A$ and $a(X_j) = B$, $d'(q, A, B) = j - i$. If q does not contain both a song from A and a song from B, then $d'(q, A, B) = \infty$. We define the *artist association degree in* q from A to B as: $w'(q, A, B) = \frac{1}{|d'(q, A, B)|}$ if $|d'(q, A, B)| \leqslant \delta'$, otherwise $w'(q, A, B) = 0$. Notice that the order is not important when we deal with artists. To estimate the *artist association degree* from any artist A to any artist B $s'(A, B)$ we use an approach similar to the one used for the song association degree using w' instead of w as in Eq. 2:

$$s'(A, B) = \frac{\sum_{q \in \mathcal{Q}} w'(q, A, B)}{f'(A)(f'(B)/\overline{f'})^{\beta}}$$

where $f'(A)$ is the number of playlists where any song by A appears (artist popularity), and $\overline{f'}$ is the average artist popularity. From the MyStrands dataset we obtain an average artist popularity $\overline{f'}$ of 235. Using $\delta' = 2$ as the maximum distance and $\alpha = 0.75$, $\beta = 0.5$, we discover that 25,881 artists have an association with some other artist.

3.1 Parameters

In the previous section, we have introduced some parameters and assigned them some specific values for our experiments and evaluations.

The value $\beta = 0.5$ was decided after several experiments, in order to obtain a nice mix of more and less popular tracks/artists in the associations. For instance, the top associated artists found for Abba when $\beta = 0.5$ are: Agnetha Faltskog, A-Teens, Chic, Gloria Gaynor, The 5th Dimension, Andy Gibb, Olivia Newton-John, Rose Royce, KC & The Sunshine Band, and The Bee Gees. Notice that

the first two names (Agnetha Faltskog and A-Teens) are not very popular, but are very much associated with Abba: the first was their lead singer, the second is a cover band of Abba. As the sequence continues, more popular names appear, still associated with Abba, but in a weaker degree.

The value $\alpha = 0.75$ was decided because we want to favour post-occurrences rather than pre-occurrences, albeit not in excess, for two reasons. The first is that it is *sometimes* useful to maintain the order of two songs (e.g., the first mixes into the second one, the first and the second are two consecutive movements of the same theme, etc.), but for some genres and songs the order is not so important. The other reason is that we assume that songs in user playlists are implicitly ordered, but this is not always the case, for some users build playlists just as unordered sets of songs.

4 Evaluation

In this section we compare the *associations* found using our system with the *degrees of similarity* provided by other community-based music services: All Music Guide, Yahoo! Music, Last.fm, MyStrands and MusicSeer[2].

All Music Guide (AMG) has handcrafted contributions by expert editors, while at Yahoo the recommendations are generated from end user feedback [4]. Last.fm similar artists focus on overall listening habits, based on people's listening history. MyStrands uses our same data, but a different technique[3], while MusicSeer follows two distinct techniques: one is based on a survey about related artists[4], the other is based on a set of playlists collected from the community Art Of The Mix.

Tables 1 and 2 compare our results for two songs with those of Yahoo! Music, the only community that deals with song-to-song similarity. Tables 3 and 4 show a comparison of our artist association model with the "most similar artists" provided by MyStrands, AMG, Last.fm and Yahoo! Music. The four artists compared are Abba, John Williams, Destiny's Child, and Frank Sinatra. When available, the results of MusicSeer are also reported.

Which observations can be done from this examples? First, we point out that All Music Guide can be seen as the base referrer, for its associations are compiled by hands by a group of experts. Nevertheless, we can observe how other automatically-compiled results are not so different from human-compiled ones; for instance our technique, MyStrands, Yahoo! Music and Last.fm, all return Dean Martin as the top result for Frank Sinatra. Also notice that our technique cannot be directly compared with other ones, because we are not looking for "similarity" but for musical association. In our case, for instance, the order of songs in a mined playlist is important, while it is not important in the case of MusicSeer playlists analysis.

[2] Their Web pages are respectively: http://allmusic.com, http://music.yahoo.com, http://last.fm, http://mystrands.com, and http://musicseer.org.

[3] For details: http://blog.recommenders06.com/wp-content/uploads/2006/09/shur2.pdf

[4] For details: http://www.musicseer.org/projects/musicsim/musicseer.org/results/

Table 1. Comparison of associated songs for *Strangers In The Night* (Frank Sinatra).

Poolcasting	*Up, Up and Away* (The 5th Dimension), *Message To Michael* (Dionne Warwick), *Whatever Happens, I Love You* (Morrissey), *Sugar Baby Love* (Rubettes), *Move It On Over* (Ray Charles), *It Serves You Right To Suffer* (John Lee Hooker), *Blue Angel* (Roy Orbison), *I Am What I Am* (Shirley Bassey), *Rain* (Jose Feliciano)
Yahoo! Music	*Mr. Tambourine Man* (The Byrds), *Don't You Want Me* (Human League), *I'm A Believer* (The Monkees), *Good Vibrations* (The Beach Boys), *Stay* (Shakespeare's Sister), *The House Of The Rising Sun* (The Animals), *Oh Pretty Woman* (Roy Orbison), *Working My Way Back To You* (The Spinners), *Never Ever* (All Saints)

Table 2. Comparison of associated songs for *Smoke on the water* (Deep Purple).

Poolcasting	*Space Truckin'* (VV.AA.), *Cold Metal* (Iggy Pop), *Iron Man* (Black Sabbath), *China Grove* (The Doobie Brothers), *Crossroads* (Eric Clapton), *Sunshine Of Your Love* (Cream), *Wild Thing* (Jimi Hendrix), *Song For Jeffrey* (The Rolling Stones), *Money For Nothing* (Dire Straits)
Yahoo! Music	*School's Out* (Alice Cooper), *Slow Ride* (Foghat), *I Can't Drive 55* (Sammy Hagar), *Rock You Like A Hurricane* (Scorpions), *White Room* (Cream), *We're An American Band* (Grand Funk Railroad), *Rock And Roll All Nite* (Kiss), *Purple Haze* (Jimi Hendrix), *All Right Now* (Free)

In general, our technique suffers less than MyStrands from the problem of over-popular artists (e.g., MyStrands returns Green Day for John Williams). Although Soundtrack is not a main genre in the dataset we have used, the associated artists we found for John Williams are closely related (indeed, most of them are other soundtrack music composers). We can observe that our technique often returns the highest association degrees for tracks/artists which are not very popular but are very strictly associated, followed by other artists which are less related but more popular. This is mainly because of the value of β, and is a desirable property for our technique to be applied in contexts where, at each moment, the next song has to be chosen from a restricted set of songs.

Finally, notice that our technique is capable to spot out (as the most associated artist) an artist which indeed is strongly associated with the previous one, although rare. For instance, our technique returns Agnetha Faltskog as the strongest associated artist with Abba, or Kelly Rowland for Destiny's Child. These are good associations, for these two are precisely the lead singers of Abba and Destiny's Child, so the relation holds. Every other music service simply ignores this kind of relationship, presenting a set of "similar artists" which are also known by the public. Our technique, on the other hand, prefers to put first in the list those artists that are *not* so famous to the generic public but that are strongly associated. This is good in the sense that the user can get to discover *new songs and artists*, not just new relations of similarity between songs/artists.

5 Social Web radio with Poolcasting

The advantage of considering *ordered sequences* of songs in our mining technique is useful when the association degrees that we discover are applied to generate good *sequences* of songs, in different contexts. Poolcasting is a social Web radio

Table 3. Comparison of associated artists for Abba.

Poolcasting	Agnetha Faltskog, A-Teens, Chic, Gloria Gaynor, The 5th Dimension, Andy Gibb, Olivia Newton-John, Rose Royce, KC & The Sunshine Band, The Bee Gees
MyStrands	Donna Summer, Madonna, Gloria Gaynor, Cyndi Lauper, Blondie, Kool & The Gang, Elton John, The B-52 s, Michael Jackson, Diana Ross
AMG	Ace of Base, Gemini, Maywood, Bananarama, Lisa Stansfield, Gary Wright, Roxette, Animotion, Clout
Yahoo! Music	The Bee Gees, The Carpenters, The Beatles, Foreigner, Whitney Houston, Madonna, Michael Jackson, Elton John, Cher, Chicago
Last.fm	The Bee Gees, Madonna, Cher, Kylie Minogue, Boney M., Michael Jackson, Gloria Gaynor, Village People, Donna Summer, Britney Spears
MusicSeer Survey	Ace of Base, Bee Gees, Blondie, Spice Girls, Olivia Newton-John, Beach Boys, Roxette, Cyndi Lauper, Backstreet Boys, Donna Summer
MusicSeer Playlists	Bee Gees, Blondie, Cyndi Lauper, Queen, Cat Stevens, Cher, Beach Boys, Donna Summer, Olivia Newton-John, Phil Collins

Table 4. Comparison of associated artists for John Williams.

Poolcasting	Williams & London Symphony Orchestra, Meco, Danny Elfman, Williams & Boston Pops, John Carpenter, London Theatre Orchestra, John Barry, Hollywood Studio Orchestra, Elmer Bernstein/RPO Pops, Spectrum
MyStrands	Danny Elfman, Vangelis, Hollywood Studio Orchestra, Erich Kunzel, Green Day, Gorillaz, Weird Al Yankovic, John Barry, Queen, Eminem
AMG	John Barry, Jerry Goldsmith, Elmer Bernstein, Howard Shore, Erich Korngold
Yahoo! Music	Patrick Doyle, James Horner, James Galway, Danny Elfman, Howard Shore, Hans Zimmer, London Symphony Orchestra, Enya, Frank Sinatra, Josh Groban
Last.fm	Howard Shore, Hans Zimmer, James Horner, Danny Elfman, Klaus Badelt, Harry Gregson-Williams, Jerry Goldsmith, Alan Silvestri, Patrick Doyle, Ennio Morricone

Table 5. Comparison of associated artists for Destiny's Child.

Poolcasting	Kelly Rowland, City High, Ciara, Fantasia, Christina Milian, Beyonce, Ashanti, Girls Aloud, 3LW, Dru Hill
MyStrands	Ciara, Pussycat Dolls, Usher, Beyonce, Nelly, 50 Cent, Mariah Carey, Chris Brown, Gwen Stefani, Eminem
AMG	Mariah Carey, Jennifer Lopez, Aaliyah, Xscape, Ginuwine, Deborah Cox, Kelly Price, Faith Evans, Brandy, Usher
Yahoo! Music	Cruel Story Of Youth, Jessica Simpson, Ryan Cabrera, Ashlee Simpson, Faith Evans, Nick Lachey, Vitaly Romanov, Janet Jackson
Last.fm	Beyoncé, Mariah Carey, Jennifer Lopez, Usher, Aaliyah, Rihanna, TLC, Ciara, Ashanti, Christina Aguilera

Table 6. Comparison of associated artists for Frank Sinatra.

Poolcasting	Dean Martin, Sammy Davis Jr., Judy Garland, Bing Crosby, The California Raisins, Tony Bennett, Louis Prima, Rosemary Clooney, Nat "King" Cole, Ella Fitzgerald
MyStrands	Dean Martin, Billie Holiday, Nat "King" Cole, Perry Como, Ella Fitzgerald, Andy Williams, Tony Bennett, Etta James, Bing Crosby, Diana Krall
AMG	Dean Martin, Vic Damone, Dick Haymes, Sarah Vaughan, Nat King Cole, Dinah Washington, Mel Tormé, Ella Fitzgerald, Tony Bennett, Jo Stafford
Yahoo! Music	Dean Martin, Tony Bennett, Bing Crosby, Nat King Cole, Elvis Presley, The Beatles, Norah Jones, Ella Fitzgerald, Louis Armstrong, Michael Bublé
Last.fm	Dean Martin, Louis Armstrong, Nat King Cole, Bing Crosby, Ella Fitzgerald, Tony Bennett, Bobby Darin, Michael Bublé, Billie Holiday, Sammy Davis, Jr.
MusicSeer Survey	Eric Clapton, Billy Joel, Elton John, Elvis Costello, Elvis Presley, Van Morrison, John Lennon, Bob Dylan, Nine Days, Ozzy Osbourne
MusicSeer Playlists	Elvis Presley, Elton John, John Denver, Abba, Whiskeytown, Beatles, Billy Joel, Bob Marley, Eric Clapton, Everly Brothers

service that automatically generates the content of radio channels using the *music pool* of the users connected to the service. The *music pool* is the collection of all songs the users have in their music players (e.g. iTunes); when a song is selected for a channel, it is uploaded to the Poolcasting server and streamed back to that channel listeners. A channel is defined by a set of conditions (e.g. genre = 'Rock' and year $>$ 1990) and a set of listening users.

In addition to the song and artist associations models that we described in the previous sections, Poolcasting takes into account the listening habits of the users, analysing the songs in their personal libraries: which tracks/artists are contained, how they were rated, how many times they were played. The scheduling of songs in a channel is determined by a Case-Based Reasoning system (CBR) that uses the association models we described as *background knowledge*, and considers each user library and its listening habits data as an *individual case base*. The goal here is not to personalise a radio for an individual person, but to customise each channel dynamically for the actual *group of listeners* at each moment in time. The CBR system is described in [3].

Poolcasting has to select one of the songs available in the music pool at each moment; for this purpose each song is evaluated combining both song association degree and artist association degree. In this combination, song association degree has more importance, but if no song-associated song can be found, Poolcasting looks for artist-associated choices. In brief, we assume that, given the last song Y played on a channel, the following songs are related and can be played on the same channel after Y (with decreasing relevance) using four layers of decision:

1. songs Z that have a strong song association degree $s(Y, Z)$ with Y;
2. songs Z that have a strong song association degree $u(Y, Z)$ with songs from the artist of Y (where $u(Y, Z)$ is the average song association degree from every song whose artist is $a(Y)$ to Z);
3. songs Z that have a strong song association degree $v(Y, Z)$ with songs from artist associated with the artist of Y (where $v(Y, Z)$ is the average song association degree from every song whose artist is associated with $a(Y)$ to Z, combined with the relative artist association degree);
4. songs Z whose artist has a strong artist association degree with the artist of Y (using artist association $s'(a(Y), a(Z))$).

Figure 1 shows an example, where a song has to be played on a channel after Abba's *Waterloo*: if a song Z can be selected using $s(Y, Z)$ at the first level, Z will be the next played song; if none can be found in the music pool, $u(Y, Z)$ is used at the second level to find Z; if not, the system proceeds to layers three and four with $v(Y, Z)$ and $s'(a(Y), a(Z))$. This intuition is implemented as the following aggregation function $r(Y, Z) = s(Y, Z) + \epsilon u(Y, Z) + \epsilon^2 v(Y, Z) + \epsilon^3 s'(a(Y), a(Z))$, where ϵ may vary in $[0, 1]$ (currently $\epsilon = 0.5$). In this way every song Z in the music pool can be assigned a relevance value $r(Y, Z)$ that represents how good it would be to schedule Z after Y.

Table 7 shows an example of the combination of these four factors to evaluate which is the best song to play on a channel after Abba's *Super Trouper*, when the music pool is made of the songs *Take On Me* (A-Ha), *Listen To Your Heart*

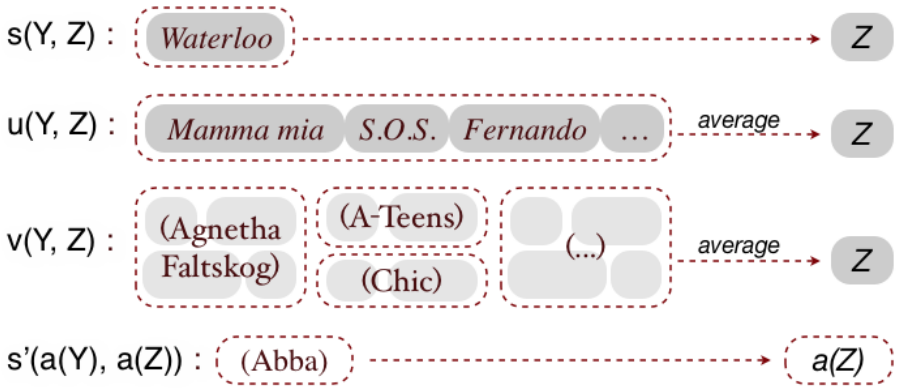

Fig. 1. Four layered factors to select next song after Abba's *Waterloo*.

(Roxette), *The Look Of Love* (ABC), and *I'm So Excited* (The Pointer Sisters).

In Poolcasting, the selection of which song to schedule on a channel in a particular moment is taken in two consecutive steps. First, $r(Y, Z)$ is used to build a list of possible candidates. Then, the music preferences of those users that are currently listening to the channel are analysed and combined, to pick the candidate that most satisfy the listeners [3]. This combination gives more weight to those users who were less satisfied with the last tracks selected to be played on that channel.

The musical preferences of each listener are inferred by Poolcasting both implicitly and explicitly. Implicitly, because Poolcasting analyses the personal music library of each participant, and infers that the higher the rating assigned to a song and the higher the play count, the stronger the preference of a user for that song. Explicitly, because with the Web interface users can evaluate the proposed songs, thus stating how much they like or dislike a specific song.

Table 7. Combining factors to find which song to schedule after Abba's *Super Trouper*.

Z_i $a(Z_i)$	$f(Z_i)$ $f(a(Z_i))$	$s(Y, Z_i)$	$u(Y, Z_i)$	$v(Y, Z_i)$	$s'(a(Y),$ $a(Z_i))$	$r(Y, Z_i)$	rank
Take On Me (A-Ha)	1341 1937	$\frac{0.942}{10^3}$	$\frac{0.574}{10^3}$	$\frac{0.324}{10^3}$	$\frac{2.817}{10^3}$	$\frac{1.662}{10^3}$	2º
Listen To Your Heart (Roxette)	184 642	$\frac{2.548}{10^3}$	$\frac{0.841}{10^3}$	$\frac{1.119}{10^3}$	$\frac{0.265}{10^3}$	$\frac{3.281}{10^3}$	1º
The Look Of Love (ABC)	237 878	0	$\frac{1.807}{10^3}$	$\frac{0.852}{10^3}$	$\frac{0.944}{10^3}$	$\frac{1.234}{10^3}$	3º
TI'm So Excited (The Pointer Sisters)	278 1149	0	$\frac{1.063}{10^3}$	$\frac{0.114}{10^3}$	$\frac{0.428}{10^3}$	$\frac{0.614}{10^3}$	4º

6 Related Work

MusicSeer endeavours to extract artist associations from user playlists. As described in [8]: "we gathered over 29,000 playlists from The Art of the Mix, a website that serves as a repository and community center for playlist hobbyists (www.artofthemix.org). After filtering for our set of 400 artists, we were left with some 23,000 lists with an average of 4.4 entries". Notice the difference in the size of experimental data between their approach and our approach.

Playlists are not the only musical data mined from the Web that has been used to discover song/artist relations. For instance [13] and [14] consider that when the *names* of two artists co-occur together in many Web pages, then they are related.

Another text-based approach is [10], that retrieves playlists from the Web in form of radio programs or users' music compilations, then tries to identify titles extracting text, in order to apply a co-occurrence analysis to assess similarity between artists. Our analysis of the occurrences of songs in playlists is different, in that we take into account the ordering of the sequences. For this reason our approach is more related to the analysis of the occurrences of words in phrases [7, 2]. In particular, we deal with sequence of songs, where the order is relevant: co-occurrences [9], and in particular post-occurrences [12].

7 Conclusions and Future Work

We have shown how knowledge discovered from a Web-based music community (song and artist association models) can be used in conjunction with individual user data to provide a customised service in the form of several automated radio channels. Poolcasting is thus an example of using data and knowledge provided from social networking for a service (automated radio channels) that is also social in nature. In this paper, we have focused on the knowledge discovery process that acquires the association models conducive to determine affinities of songs that are to be played in sequence. Poolcasting is currently being tested in our local network. We plan to deploy it to the Internet; actually this would require us to pay the copyright fees normally applied to commercial Web radios.

Our focus on sequential ordering information could be used, as we plan to do in future work, for other tasks in addition to Poolcasting. A straightforward application is a media player plug-in that generates playlists for a single user, considering her library of songs as the only music pool from which songs have to be selected. Another application is generating "channels for parties", as done in PartyStrands[5]; in this context the association models would work the same way, but we would need to change the way in which data from individual profiles are acquired.

In this paper we have shown how to calculate associations from *one* song to another; this approach could be expanded to look for contiguous sequential

[5] http://partystrands.com

patterns: sequences of two or more songs that appear frequently in the dataset of playlists and as such can be considered strongly associated.

References

1. Andric A. & Haus G. (2006). Automatic playlist generation based on tracking user's listening habits. *Multimedia Tools and Applications*, vol. 29(2), pp. 127–151.
2. Ahonen-Myka H. (1999). Finding All Frequent Maximal Sequences in Text. In *ICML-99 Workshop: Machine Learning in Text Data Analysis*, Mladenic D. & Grobelnik M. (eds.), pp. 11-17.
3. Baccigalupo C. & Plaza, E. (2007). A Case-Based Song Scheduler for Group Customised Radio. In *Proc. of the 7th Int. Conf. on Case-Based Reasoning (ICCBR '07)*, Weber R.O. & Richter M.M. (eds.), LNAI 4626, pp. 433–448.
4. Baumann S. & Hummel O. (2003). Using Cultural Metadata for Artist Recommendations. In *Proceedings of the Int. Conf. on WEB Delivering of Music (WEDELMUSIC '03)*, pp. 138–141.
5. Brown B., Sellen A. & Geelhoed G. (2001). Music Sharing as a computer supported collaborative application. In *Proc. of the 2001 European Conf. on Computer Supported Cooperative Work (ECSCW '01)*, pp. 179–198.
6. Ellis D.P.W., Whitman B., Berenzweig A. & Lawrence S. (2002). The Quest for Ground Truth in Musical Artist Similarity. In *Proceedings of the 3rd Int. Symposium on Music Information Retrieval (ISMIR '02)*, pp. 13–17.
7. Heylighen F. (2001). Mining Associative Meanings from the Web: from Word Disambiguation to the Global Brain. In *Proceedings of the International Colloquium: Trends in Special Language & Language Technology*, Temmerman R. & Lutjeharms M. (eds.), pp. 15–44.
8. Logan B., Ellis D.P.W. & Berenzweig A. (2003). Toward Evaluation Techniques for Music Similarity. In *Proceedings of the 4th Int. Symposium on Music Information Retrieval (ISMIR '03)*, pp. 81–85.
9. Mannila H., Toivonen H. & Verkamo A.I. (1995). Discovery of Frequent Episodes in Event Sequences. *Data Mining and Knowledge Discovery*, vol. 1(3), pp. 259-289.
10. Pachet F., Westermann G. & Laigre D. (2001). Musical Data Mining for Electronic Music Distribution. In *Proceedings of the Int. Conf. on WEB Delivering of Music (WEDELMUSIC '01)*, pp. 101–106.
11. Schedl M., Knees P. & Widmer G. (2005) Discovering and Visualizing Prototypical Artists by Web-based Co-occurrence Analysis. In *Proceedings of the 6th Int. Symposium on Music Information Retrieval (ISMIR '05)*, pp. 21–28.
12. Sudkamp T. (2004). Co-occurrence, interest, and fuzzy events. In *Fuzzy Information, 2004. Processing NAFIPS*, vol.2, pp. 508–513.
13. Whitman B. & Lawrence S. (2002). Inferring Descriptions and Similarity for Music from Community Metadata. In *Proceedings of the 2002 Int. Computer Music Conf. (ICMC '02)*, pp. 591–598.
14. Zadel M. & Fujinaga I. (2004). Web Services for Music Information Retrieval. In *Proceedings of the 5th Int. Symposium on Music Information Retrieval (ISMIR '04)*, pp. 478-483.