Programme driven music radio

Conor Hayes, Pádraig Cunningham, Patrick Clerkin, Marco Grimaldi
Department of Computer Science
Trinity College Dublin

{conor.hayes,padraig.cunningham@cs.tcd.ie}

Abstract. This paper describes the operation of and research behind a networked application for the delivery of personalised streams of music at Trinity College Dublin. Smart Radio is a web based client-server application that uses streaming audio technology and recommendation techniques to allow users build, manage and share music programmes. While it is generally acknowledged that music distribution over the web will dramatically change how the music industry operates, there are few prototypes available to demonstrate how this could work in a regulated way. The Smart Radio approach is to have people manage their music resources by putting together personalised music programmes. These programmes can then be recommended to other listeners using a combination of collaborative and content-based recommendation strategies. We describe how we use a novel two-stage approach to find recommendations that are pertinent to a listener's current listening preferences, something which collaborative techniques are insensitive to. We describe additional constraints required to provide a service personalized to each user's listening preferences. The Smart Radio system currently runs within the Computer Science Intranet with permission from the National Music Rights Organisation It is a prototype system for an "always on" high bandwidth Internet connection such as ADSL.

1 INTRODUCTION

The ease with which high quality digitally compressed music can be distributed on the Internet has caused waves in the music industry. Whereas the sale of music has been a highly regulated activity, the Internet has posed many problems and new opportunities. Since mp3 music items can be downloaded quickly by large numbers of people, the piracy and copyright infringements suffered are much greater than in days when the industry would warn that "home taping is killing music". However, the music industry has generally been slow to capitalise on the benefits of this new distribution channel. Much of this has to do with the lack of agreement on standards for digital authentication, distribution methods and payment models that would be attractive to consumers. Some consensus has emerged recently with several companies now offering streaming services for a monthly subscription.[1]

The advent of online music services poses similar problems of information overload often described for online textual material. However, the filtering of music resources has its own peculiarities. This paper describes a personalised web-based music service called Smart Radio, which has been in operation in the computer science department at Trinity College Dublin for the past two years. The service was set up to examine how a personalised service of radio programming could be achieved over the web. Section 2 describes the system operation and architecture. We introduce the idea of a programme, a user-compiled collection of music tracks that we use as the basic unit of recommendation. Section 3 introduces the general methodologies used for personalisation, namely Automated Collaborative Filtering (ACF) and Case-Based Recommending. Section 4 elaborates the deficiencies of using ACF as the sole recommendation strategy, and introduces the idea of a *context*, a strategy we use to further refine recommendations made by the ACF engine. In section 5, we discuss other constraints required to successfully recommend programmes. Finally, in section 6, we identify weaknesses in the current set-up and describe current work on improving the content-based recommendation strategy.

2 SYSTEM OPERATION AND ARCHITECTURE

2.1 Programmes

Smart Radio is a web based client-server application that allows its users to listen to their favourite music and receive new music while connected to the Internet. Users login through their browser, after which they can build new programmes from scratch, play past favourite programmes, or choose programmes recommended by their neighbours.

The service paradigm we sought to emulate was that of a personalised radio service centred on the user and other listeners of similar taste. Individual listeners put together programmes of music that are then recommended to other similar minded members.

Since music radio is a well-understood concept, by tying together the concept of a music programme with a personalised recommendation system, we sought to provide an intuitive stream of music service. By using a programme of music as our unit of recommendation the work involved in always putting together a new compilation of music is bypassed.

Another benefit is that a collection of music may contain the difficult to quantify feature of "mood" which depends on the collected items being played together. This feature is apparent where users amend their ratings for individual items as they appear in different programmes. Rather than making anonymous suggestions, Smart Radio allows the listener to identify who authored each recommended programme. This facility encourages community participation by allowing users know who their most consistent neighbours are. In this way, a recommendation system promotes rather than replaces social processes [1].

2.2 Streaming technology

We chose to stream audio files to our listeners' desktops rather than allow them download the individual files for a number of reasons. If the basic unit of recommendation were an mp3 file, the time involved in downloading several of these to a hard drive and then setting up a play-list would be quite considerable. With the streaming model used by Smart Radio, a user can login, choose a recommended programme or a favourite programme, and have a

[1] www.pressplay.com
www.musicnet.com
www.higherwaves.com

music programme stream to the desktop almost immediately. Apart from the benefits of immediate delivery, we considered streaming technology to be a useful solution to the piracy issue. The streamed audio files are not saved to the user's hard drive and thus cannot be posted on the Internet for illicit distribution. Another benefit of a service where there are no downloadable files is that the user has a roaming profile and his music programmes and recommendations are available wherever he logs on. Though disk space has become cheaper, there is considerable management involved in holding a large digital library on a local machine.

2.3 Operation and Feedback

Figure 1. A programme in Smart Radio. The icons to the right indicate the user's rating of the tracks in the programme.

Listeners may search the music database by artist name, song title or genre. They can add a music item to the current programme with one click of a mouse. Indexing music items by genre is not very satisfactory since the genre feature is not finely grained enough to capture the nuances within music types. Section 6 describes research we are currently undertaking to capture more representative features for better indexing. Once a programme contains ten music items it can be played immediately and is automatically saved to the user's profile for future retrieval.

Figure 1 illustrates a programme in which the user *conorc* has rated all ten items. If *conorc* chooses this programme again he will be shown his ratings for the individual items within the programme and he may recast his vote. This facility is important because music taste does shift, and user profiles will have to move to reflect this.

In order to provide data for our recommendation engine we collect explicit and implicit feedback from our listeners.

Users can provide explicit ratings on individual track items or individual programmes on a scale of 1– 5, where 5 is the top score. It is important to infer feedback, particularly for the less interactive or casual user [2]. This involves monitoring the use-data for each user and allocating positive ratings to frequently played items.

Unlike the user *conorc*, the casual user may choose to interact minimally with the system intervening only when the music provided is not to his taste.

2.4 Architecture

Smart Radio is a client-server system. The server components consist of a web server with java servlet extensions and a recommendation engine, which consists of an ACF engine, a CBR engine and a module that evaluates user preferences (see Fig. 2).

The client side consists of a browser with a plug-in component for playing streaming audio files. A streaming audio server runs in parallel with the web server. This server receives requests from the clients and streams audio pieces to the specified address. The Smart Radio web pages are able to play the incoming stream by means of a streaming audio plug-in embedded in a HTML frame which remains constantly in place irrespective of what other pages the user requests from the Smart Radio server. In fact it is this plug-in that requests the file containing the addresses of constituent pieces of the programme selected by the user.

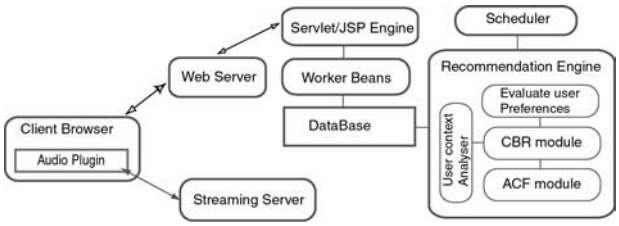

Figure 2. Smart radio architecture

3 ACF AND CASE BASE USER PROFILING

Perhaps the obvious way to implement a system for recommending assets to a customer is to work with representations of the assets and the user's interests and use these to match assets to users. This is commonly referred to as content-based or case-based recommendation and two sample cases for recommending music are shown in Table 1. This representation contains some evidence that this user might like this track. While this approach has the advantage of simplicity, it has the drawback of needing to mark up assets and users in the appropriate representation and also the problem of coming up with the appropriate representation in the first place. This representation problem is particularly acute in music recommendation where descriptors such as 'genre' are inclined to be over-burdened.

Table 1. A representation of a music track and a profile of a user that might be used in content-based recommendation.

TB-2	
Title	Unbreak My Heart
Year	1996
Genre	Pop, Soul
Artist	Toni Braxton
Album	Secrets

JB-7	
Name	Joe Bloggs
Preferred Era	1990 +
Genre	Soul, RnB, Pop
Fav-Artists	Lauryn Hill, Macy Gray George Michael
Fav-songs	M117, M144, M56, M89
Fav-albums	A232, A200, A401, A212

The ACF (Automatic Collaborative Filtering) approach to recommendation finesses this representation problem by basing recommendations on users' ratings of assets only and uses no descriptions of assets or users' interests. An example of such a representation is shown in Table 2 which shows that User 1 has given a rating of 0.8 (4 out of 5) to Track 1, no rating to Track 2 and so forth. The basic idea behind ACF involves identifying users with shared interests and making recommendations based on this. In the example in Table 2, Track 2 might be recommended to User 1 based on the similar ratings for Tracks 2, 3 & 4 by Users 1 and 2. (The details on how recommendations are generated are given in [3,4])

Table 2. The type of representation used in ACF where a user's profile is simply a list of ratings for tracks.

	Track 1	Track 2	Track 3	Track 4	…
User 1	0.8		0.2	0.8	
User 2	0.6	0.8	0.2	0.4	
User 3	0.4	0.8		0.2	
User 4		0.2	0.8		

Given sufficient data, the ACF approach can produce excellent recommendations. However it is not without its shortcomings, the main ones being the problem of handling new assets and users and the absence of any temporal model of users interests. This last issue is a particular problem in recommending music where a user's listening interests may vary over time. The way this is handled in Smart Radio is described in the next section.

3.1 ACF in Smart Radio: recommending programmes

Each listener in Smart Radio has a listening history consisting of a number of programmes with their component tracks. The listener may have explicitly rated individual tracks in each programme. Programmes may also have been repeatedly played or contain overlapping tracks from which we can implicitly infer positive feedback for component tracks. From this we build the type of table illustrated by Table 2. Correlations are made between listeners based on the component tracks in their programme history rather than on the programmes themselves. Two listeners may be highly correlated based on the overall component tracks they have listened to, even though they may not have listened to the same programmes. In typical ACF fashion a neighbourhood is then established for each user, and programmes rated highly by these neighbours are retrieved.

Making a prediction on a programme involves making a prediction for each of the component tracks where we don't have any explicit or implicit feedback. The overall prediction for a programme is a weighted sum of the prediction for each the component tracks divided by the number of tracks in the list. The weighting refers to a user–defined constraint that we call the *novelty factor*. A criticism often levelled at version 1 of Smart Radio was that it didn't take into account user preferences for receiving new music.

Listeners may specify on a scale of 1 to 5 the degree of novelty they want in their recommended lists where 1 signals the user's preference for programmes with a large amount of already known music. This caters for listeners who do not want programmes that contain very much music they do not already recognise.

Table 3 illustrates a programme where we have made predictions for five of the component tracks. The status field refers to whether a score is known for a particular track (k), has to be predicted (p) or is unknown (u). In the case of u, the score allocated is the average rating for the track.

Table 3. illustrates a programme where we have 3 items already rated by the user (k), 6 items for which we have made predictions using the listener's nearest neighbours (p) and one for which we were unable to make a prediction (u).

Track Id	Status k/p	score
127	k	0.8
23	p	0.6
45	k	0.6
78	p	0.8
206	p	0.6
1266	p	0.6
587	p	1.0
13	p	0.8
234	k	0.8
1500	u	0.4

Equation 1 gives the overall score, s, for any programme where *w* refers to the *novelty factor* and *n* is the number of tracks in the list.

$$1) \qquad s = \frac{(1-w)\sum k_i + w(\sum p_i + \sum u_i)}{n}$$

Using this Equation the score for this programme where the listener has specified a low novelty factor of 0.25 is 0.27.

```
(0.75(0.8+0.6+0.8) +
0.25(0.6+0.8+0.6+0.6+1.0+0.8+0.4))/10 = 0.285
```

The score for the programme where there is a relatively high novelty factor of 0.75 is 0.43.

```
(0.25(0.8+0.6+0.8) +
0.75(0.6+0.8+0.6+0.6+1.0+0.8+0.4))/10 = 0.415
```

The Programmes are then ranked according to these scores. The following section describes a refinement to this first stage retrieval that seeks to promote programmes that are most pertinent to the listener's current listening behaviour.

3.2 Recommendations

Recommendations are calculated periodically and in response to listener feedback. The recommendation engine maintains a correlation matrix in memory that is updated every half hour for the users that have been online since the last update. This is a triangular matrix since the ACF correlation function is commutative. Using the correlation scores in the matrix and the listening context (a feature explained in section 4.) the system updates the database with the most current recommendations.

However, in response to a substantial amount of new information (feedback) being introduced into the system by a listener, the recommendation engine updates the matrix row associated with the listener and his listening context. We use a rule of thumb to trigger these recommendation threads: if a listener explicitly changes his profile by 20% we immediately recalculate his correlations and listening context. This strategy tends to support new users who would not have a lot of data in the system in the first place.

4 CONTEXTS

ACF is a successful methodology for managing the long-term resource requirements of the online user. The user's interaction with an ACF based system is usually of a sustained nature involving a dialogue that may last from a few minutes to a few years in the case of a successful retail portal like Amazon.com. [4]. The basic methodology involves lazily making recommendations using the full user history. The sequence of items played most recently does not influence the recommendation strategy of the basic ACF algorithm. However not all the data informing a user's recommendations may be pertinent to the current listening preferences. For instance, a listener may listen to rock'n'roll, jazz and dance music but may wish the system to learn his current listening preference by recommending suitable jazz oriented programmes when he is listening to jazz. We see two possible ways of addressing this. Firstly, we might use only a portion of the user's profile and then make ACF recommendations using this reduced profile. Isolating the subset of items that are pertinent to the current listening context may be quite difficult, particularly in a

domain where there is not much content description. Secondly, there is no guarantee that our listener will receive Jazz recommendations if his nearest neighbours have as eclectic a taste as himself.

The approach currently employed by Smart Radio involved the development of **a local context analyser.** This module was developed to cater for personalisation requirements that are context specific. This module maintains a simple user profile that summarises the listener's most recent listening patterns. This is in effect a *sliding window* on the user's listening history where the window is the last *n* programmes. This relies upon having some representation of the assets being recommended. In the case of the music items in Smart Radio, we make use of features freely available in the ID3 tag of the mp3 file. Most mp3 ripping software includes this information in the last few bytes of the mp3 file itself[2]. While these features are not highly predictive on their own they give us a good indication of the user's current preferences.

Our approach is to use the full user profile for ACF recommendations but to then refine these recommendations based on similarity to the current listening context. This two-fold strategy is a type of MAC/FAC retrieval well known amongst CBR researchers [5]. Our case base consists of the total number of programmes in the system. The ACF module is responsible for the first stage retrieval (MAC) while refinement to this retrieval is carried out by similarity-based retrieval (FAC). Implementation wise, the second stage of this retrieval is carried out using a spreading-activation net where each programme is treated as a case in the case memory (the activation net). In the first stage, the ACF module primes a subset of the case memory. In the second stage, the user-profile is presented as a target case. Activation spreads only through the primed subset of the case memory. Those programmes that have highest activation after this process are presented as recommendations to the listener.

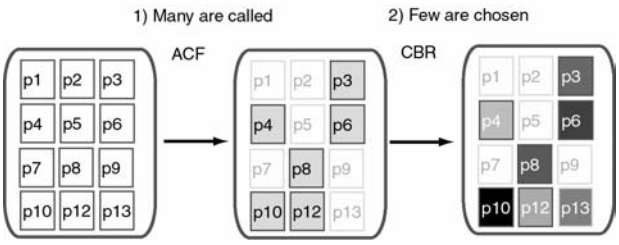

Figure 3. The darker shaded cases in the third stage indicate cases which best match the listener's current listening context.

The profile provides a means of scoring recommended programmes according to their similarity to the sliding profile. The recommendations are then sorted based on these scorings.

This ensures that programmes created and endorsed by the listener's nearest neighbours but which also best match the user's current listening preferences are pushed to the top of the heap. The sliding profile does not filter out neighbours' programmes that do not match the current context since it is understood that the listener may well wish to break out of the current vein of music.

4.1 Negative profiles

The context builder also allows negative user profiles to be built. These are profiles are constructed based on negative feedback issued by listeners. Unlike the sliding window profiles these profiles are built using the full listening history of the user. Their employment is user-policy based in that the user has access to the negative profile and may choose to have it used or not. In the case where it is put into effect, the negative profile acts as a filter removing programmes that have high similarity to the negative profile. This stage is carried out directly after the ACF retrieval and prior to the positive profile refinement stage. The user can determine the number of negatively rated artists or items a programme may have before it is deemed unacceptable. In the next section we introduce additional policy based features that allow our listeners personalize their listening experience.

4.2 Similarity measures

The weak representation of our music assets obviously poses some problems. In table 1 we have illustrated a typical user profile and a case-like representation of a music item. Defining similarity measures for some of the features shown requires a certain amount of expert knowledge. In the case of the *genre* feature we have defined a similarity measure relating certain genre types such as blues, jazz and r'n'b. For other genre types we just haven't enough insight into the problem to specify a similarity measure – in which case the similarity value is either one or zero. Work has been carried out on data-mining Smart Radio user data to find higher level concepts that would describe our music assets and provide us with similarity measures based on our listener's usage patterns [6,7]. These concepts would constitute a new type of genre information that should reflect our listeners' view of the music assets in the system. This work is described in section 6.

Maintaining a case base of programmes in memory has a secondary benefit. It also allows the user to query the system using the current programme as a target. This, in effect asks the system to retrieve a programme similar to the current one. However, with the rather weak features currently available to us, the success of this approach is limited. Section 6 will discuss, how improved features will benefit the system.

5 ADDITIONAL CONSTRAINTS

In this section we draw a distinction between a personalization service and a recommender service. We introduce some user-guided constraints that allow each listener to affect how the recommendation engine provides new programmes.

We view Smart Radio as a personalization service that allows people to manage their listening habits. As such it seeks to model our users' listening patterns and anticipate their audio requirements. Typically recommender services will provide only new items with the assumption that once a recommended item has been utilized, it should not be recommended again.

However, the personalization service offered by Smart Radio recognises that music can be played over and over again. It is a normal feature of the programme format of music radio that popular artists and songs are played quite frequently.

In section 3.1 we described the novelty constraint that allows users to state their preference for the amount of new music they receive in recommended programme. Some users prefer not to receive a lot of new music in a listening session. In order to prevent the same tracks turning up in successive programmes we allow the user to specify a minimum period before an individual track can be played again. Real radio stations are guided by broadcasting regulations in this respect. In our situation, the user can set the minimum period. This is implemented by assigning user-music item pairs an *excitation level*. Excitation is high immediately after an item is played, but decays with time so that a music item can be played again when sufficiently low. The decay function is based on the user preference for the minimum period for replaying a piece of

[2] http://www.id3.org/

music. Programmes therefore are assigned an overall excitation level based on component tracks. This score is considered when making a final set of recommendations. Obviously, in this case programmes with low excitation levels are preferred.

6 FURTHER WORK

The poor representations currently available of our music assets mean that our similarity-based strategy is the weak link in the two-fold retrieval outlined in section 4. The problem has been addressed from two angles. The first seeks to elaborate our content descriptions using knowledge mined from our listeners use data. Our goal is to produce an ontology that codifies the relationships between music items based on our listeners' listening data. The usual way of creating an ontology is for domain experts to establish the fundamental concepts, objects, relations, etc, which exist for a given community. This presupposes that these ontological elements can be uncovered *a priori*. However, in domains such as that of Smart Radio, it is not at all clear that any *a priori* analysis by a team of experts could yield the sort of concepts important to recommendation tasks. While songs may be categorised according to artist, and while to a much lesser extent, genres and sub-genres may be employed, this approach is inadequate, since it does not account for the fact that many people like songs that are widely divergent according to the artist and genre criteria.

In pursuit of this, we have used Fisher's COBWEB algorithm [7] to produce a hierarchical classification scheme for songs in the system [8]. Each music item is represented as an object with features corresponding to each of the users in the system and with values corresponding to the rating assigned to the song by that user. By using such algorithms as COBWEB to cluster songs based on user ratings, we are endeavouring to discover structures more truly reflective of the similarities and dissimilarities between songs. We need only evaluate the resulting conceptual structures in terms of their impact on recommendations, and we need not worry that users may be unable to articulate the perceived similarities and dissimilarities that are hypothesised between songs. Furthermore, we do not expect that the discovered conceptual hierarchy will map onto any existing and already familiar network of human concepts. Rather, we expect to discover structures that human analysis has not detected. A further advantage conferred by the automatic generation of ontologies using COBWEB is that such concept formation algorithms are *incremental*, in the sense that observations are not processed *en masse*. There is a stream of objects, which is processed over time. In the case of Smart Radio, this means that the conceptual hierarchy can automatically evolve over time as new songs are added to the database, and as new users join the system. The capacity to evolve over time is essential to a constantly expanding online community resource. The type of concept hierarchy we describe would allow us to calculate a similarity between songs. Similar songs will fall under the same concept and degrees of similarity will be captured in the relationships between concepts.

The second way in which we are seeking to improve the representation of our assets involves automatic feature extraction from audio files. We propose to use rhythmical features and features from frequency analysis that will allow us to identify instrumentation. It has been shown in [9] that nine parameters characterise the strength of the beat and that rhythmical features may be used with good results for music classification purposes.

Since instruments are characterised by a typical range of frequencies, by analysing the frequencies being played it may possible to distinguish between simple polyphonic music such as blues and rock and more complex music compositions in the fields of jazz and classical music. Having music assets characterised in terms of rhythm and instrumentation would improve our similarity retrieval strategy greatly. We would, for instance, using the rhythm feature alone be able to differentiate fast programmes from slow programmes of music. This would obviously be helpful in finding a programme suited to the listener's current context. We would also be able to search for a programme that best matches the rhythm profile of the target programme.

7 CONCLUSION

Smart Radio is client-server application that provides its users with personalised service of streaming music in the form of programmes built by other listeners. A MAC/FAC retrieval strategy is employed to recommend new programmes to the listener. This works by having an ACF module retrieve programmes suited to the target listener followed by similarity-based refinement which sorts these programmes according to the listeners' current listening context. We introduce the idea of a negative profile, which is a summary of the music items disliked by the user. The user determines the deployment of this profile. Likewise, we introduce two user-defined restrictions: the *novelty factor* and *excitation level*. The novelty factor allows listeners to specify their preference for receiving new music. This is implemented as a weight on each component track when making a prediction for a particular programme during first stage retrieval. The excitation level is a means of promoting programmes that contain tracks that have not been played recently. We discuss how the poor representations of our music items mean that our similarity-based retrieval is the weak link in our system. This strategy may be augmented by information derived from knowledge discovery techniques. We finally introduce work we are carrying out on automatic feature extraction from audio files to improve our content-based recommendation.

REFERENCES

[1] Hill, W., L., Rosenstein, M., and Furmas, G. (1995) Recommending and Evaluating Choices in a Virtual Community of Use. In proceedings of the conference on human factors in computing Systems (CHI95), 194-201, Denver, CO, ACM Press

[2] Loeb, Shoshana (1992) Architecting Personalized Delivery of Multimedia Information. CACM/December 1992/Vol.35. No12

[3] Hayes, C., Cunningham, P. (2001) SmartRadio – community based music radio; Knowledge Based Systems, special issue ES2000, Volume 14, Issue3-4, June 2001, Elsevier

[4] Hayes, C., Cunningham, P., Smyth, B., (2001) A Case-Based Reasoning View of Automated Collaborative Filtering, in Proceedings of 4th International Conference on Case-Based Reasoning eds D. W. Aha, I. Watson, LNAI 2080, pp234-248, Springer Verlag.

[5] Gentner, D., and Forbus, K. D. 1991. MAC/FAC: A model of similarity based access and mapping. In Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society. Erlbaum

[6] Clerkin, P., Hayes, C., Cunningham, P (2001) Automated Case Generation for Recommender Systems Using Knowledge discovery techniques. (2001) Proceedings of Workshop on Case Based Reasoning in Electronic Commerce at the Fourth International Conference on Case-Based Reasoning 2001, Vancouver BC, Canada.

[7] Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. Machine Learning, 2, 139-172.

[8] Clerkin, P., Cunningham P., Hayes C. (2001) Ontology Discovery for the Semantic Web Using Hierarchical Clustering in proceedings of Semanctic Web Mining Workshop at ECML/PKDD-2001, September 3, 2001, Freiburg, Germany

[9] G. Tzanetakis, G. Essl, P. Cook, "Automatic Music Genre Classification of Audio Signals", In Proceedings of the International Symposium on Music Information Retrieval (ISMIR), Bloomington, Indiana, 2000