Context Boosting Collaborative Recommendations

Conor Hayes, Pádraig Cunningham
Computer Science Department,
Trinity College Dublin

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. Since good content descriptors are difficult to obtain in the audio domain, we originally used automated collaborative filtering, a 'content less' approach as our recommendation strategy. We describe how we improve the ACF technique by leveraging a light content-based technique that attempts to capture the user's current listening 'context'. This involves a two stage retrieval process where ACF recommendations are ranked according to the user's current interests. Finally, we demonstrate a novel online evaluation strategy that pits the ACF strategy against the context-boosted strategy in a real time competition.

1. Introduction

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. The advent of on-line music services poses similar problems of information overload often described for textual material. However, the filtering/recommendation of audio resources has its own difficulties. Chief amongst these is the absence of good content description required by content or knowledge based systems. This drawback is conventionally overcome using collaborative filtering, a technique that leverages similarity between users to make recommendations. As such it is often termed a 'contentless' approach to recommendation because description of the items being recommended is not required. Apart from the obvious advantage of a 'knowledge–light' approach to recommendation, ACF is often attributed as being able find recommendations that would otherwise escape content based recommender strategies. This is because it relies upon user preferences that may capture subtle qualities such as aesthetic merit that may escape current content-based systems. However, ACF does have well documented drawbacks such as the problem of bootstrapping new users and new content into the system. In this paper we examine a less documented weakness, that of *context insensitivity*, and provide a solution using a light-weight case-based approach. Since our technique imposes a ranking based on what the user is currently listening to in the system we do not consider offline approaches to evaluation such as cross validation or measures of recall/precision appropriate for this situation. Instead we measure whether a user was inclined to make use of the recommendations presented to them. We evaluate

our approach using a novel on-line methodology in which a pure ACF strategy and a context boosted ACF strategy are concurrently deployed. We measure how well both techniques perform and find that context-boosted ACF significantly outperforms ACF. Section 2 briefly describes the system operation, and introduces the idea of a playlist, a user-compiled collection of music tracks that we use as the basic unit of recommendation. In section 3 we introduce some of the principles of Automated Collaborative filtering (ACF). We point out some of the deficiencies of using ACF as the sole recommendation strategy, and in section 4 we introduce the idea of a *context* and a strategy we use to further refine recommendations made by the ACF engine. Section 5 describes the integration of our hybrid approach. Finally, in section 6 we discuss our choice of evaluation methodology and present our results.

2. Smart Radio

Smart Radio is a web-based client-server application that allows users to build compilations of music that can be streamed to the desktop. The idea behind Smart Radio is to encourage the sharing of music programmes using automated recommendation techniques. As well as this Smart Radio users can always see their neighbourhood profile, and can elect to receive new programmes from neighbours they learn to trust. The unit of recommendation in Smart Radio is the *playlist,* a compilation of music tracks built by one listener and recommended to other like-minded listeners. In terms of music delivery it is a better-understood format than a single song recommendation system. It has the advantage of allowing a full programme of music to be delivered to the user. In this way, the *work* involved in compiling a playlist of music is distributed to other listeners. The playlist format is also attractive in that it allows individual users to personalise their own selections of music, with the understanding that these will be passed on to like-minded listeners. This is often reflected in the titles Smart Radio listeners give to their playlists. Our original hypothesis was that the playlist format also would also capture the implicit 'rules of thumb' that people may use when putting together selections of music, such as "don't mix techno and country". However, this hypothesis has yet to be fully tested.

Figure 1. The Smart Radio recommendation panel. Recommended playlists are presented as an ordered list. By default the first playlist is automatically displayed.

3. ACF and its Drawbacks

The first version of Smart Radio used Automated Collaborative Filtering (ACF) as its recommendation strategy (Hayes & Cunningham 2000). ACF does not attempt to model any aspect of cognitive psychology as case based reasoning does. Instead it relies upon implicitly capturing the subtle distinctions people make when selecting or rejecting from any set of items. Its great strength is that it operates without using any representations of the items being filtered, relying instead on the pooled preference data from positively correlated users to recommend or reject from the resources available. As such it has been referred to as 'word of mouth' filtering (Shardanand and Maes 1995), 'people to people correlation' (Schafer et al. 1999) and a 'contentless' approach to filtering or recommendation (Resnick et al. 1994, Balabanovic & Shoham 1997); Even more so than CBR, it is used to as a recommendation strategy where the knowledge acquisition bottleneck is severe.

	item A	item B	item C	item D	item E	item F	item G
User 1	0.6	0.6	0.8	?	?	0.8	0.5
User 2	?	0.8	0.8	0.3	0.7	?	?
User 3	0.6	0.6	0.3	0.5	?	0.7	0.5
User 4	?	?	?	?	0.7	0.8	0.7

Table 1. The table illustrates user-preference data the type of data used in ACF systems. Although this type of data is ordered, this ordering is not used in the ACF algorithm.

One serious drawback to ACF is that it is not sensitive to a user's interests at a particular moment. Even though a user's preference data is an ordered set of ratings collected over time, the data is treated in an *accumulative* fashion by the ACF algorithm. The sparsity of the data necessitates taking all ratings into account in order to make sound correlations with other users. However, the resulting recommendation set will contain a mix of items reflecting each user's accrued interests. This may not be a real drawback if we are using ACF as a passive filter. However, where ACF is required to produce recommendations on demand, its lack of sensitivity to the users current interests may cause frustration and distrust. For instance if a user is engaged in specialised activity, such as reading documents on a particular subject, or listening to music of a particular genre many of the recommendations will be inappropriate and untimely. The problem is complicated by the fact that many ACF recommender systems operate in domains where there is very little content description, making it difficult to ascertain the transitions between interests of a particular type As we illustrate in figure 2, our goal is to enhance the usefulness of an ACF based system by using a lightweight content-based strategy like CBR to rank ACF recommendations according to the user's current interests. The darker shaded cases in the diagram indicate ACF recommendations that are most similar to the user's current *context*. In the next section we explain the concept of *context*, a representation of the user's interests at a particular moment and we present our case-based representation of playlist context. In section 5 we present our hybrid ACF system which uses a novel MAC/FAC retrieval strategy.

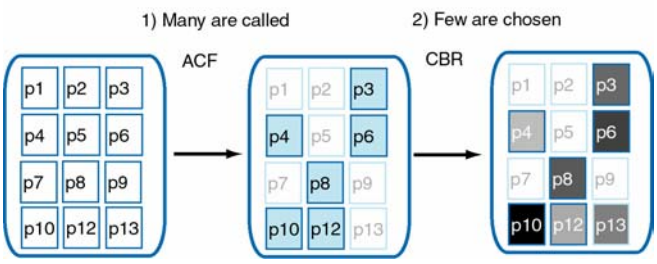

Figure 2. The ACF module selects a subset of the case base. A second stage retrieval process then ranks these primed cases according to their similarity to the user's listening context. This is indicated by the darker shaded cases on the right.

4. Context boosted ACF

This concept of isolating localised interest has been referred to in user-modelling research as *context*. It is a slippery term that has a wider variety of meanings as it is also used to describe sets of environmental parameters in the area of ubiquitous computing (Schlit et al 1994). Our use of the term is similar to what would be termed 'task context' within the same community. This is a description of the explicit goals, tasks, actions, background and activities of the user at a particular time and place. The objective to isolating context in user-modelling is that tasks being undertaken by the user may be anticipated and a portion of the work carried out automatically in advance. An example of such a technique is *Watson* (Budzik et al. 1998) an application which monitors the user's current writing or reading behaviour on desktop applications such as Microsoft Word or Windows Explorer, and using information retrieval techniques for content analyses, automatically issues requests to search engines in order to retrieve related material. Another such is Letizia (Liebermann 1995, Liebermann et al. 2001), an application that tries to predict the most relevant links the user should explore based on the pages he or she has viewed in the previous session. Letizia operates concurrently while the user is browsing, building a keyword based user profile of the current browsing session. It downloads linked pages and generates a score for each link based on the user-profile. Letizia presents a recommendation set of links that indicate a preference ranking according to the current state of the profile. The objective is to recommend a certain *percentage* of the currently available links. Both Watson and Leitizia have been termed 'Reconnaisance' applications (Liebermann et al. 2001). In both cases the user-profile is a *short-term* representation of the user's current interests designed to capture the context of the current task undertaken by the user. The context is a content-based representation of items currently being used. This can be viewed as an approximation of a task-based representation where the user's explicit task goals are known. Obviously the approximation is noisy because it is based upon an implicit concept of the user's interests. If the user digresses or switches subject while researching a topic, both reconnaissance aides will require time to respond. However, the advantage of an implicitly generated profile is that the user does not need explicitly to describe his/her goals, prior to working. Measuring the success of the short-term user profile is a difficult issue. The

problem boils down to analysing the correctness of the ranking produced according to their relevance to the user profile. Whereas analyses of recommender systems have been reliant on off-line machine learning evaluation, ranking problems such as these are not as easily studied in on offline manner. In section 6 we present an evaluation technique suited to measuring the success of contextually motivated recommendations.

4.1 Contextualising by instance

The primary recommendation strategy in Smart Radio is ACF. However, ACF is not sensitive to the user's current interests. So while the user is listening to jazz music, he/she is as likely to receive an electronica playlist as he is a jazz playlist. The objective to context-guided ACF is to recommend items based on neighbour endorsement as before, but to promote those items that may be of interest to the user based on his/her current context. Unlike the examples of the *reconnaissance* aides described earlier, which used information retrieval analyses to build a short-term user profile, the Smart Radio domain suffers from a deficit of content descriptions. Our goal is to enhance the ACF technique where very little content is freely available, and where the knowledge engineering overhead is kept to a minimum. The content descriptors we use are found in a few bytes of information at the end of the mp3 file. The type of information available is *TrackName, artistName, albumName, genre* and *date.* However, since this information is often voluntarily uploaded to sites such as the CD database (www.cddb.com), track information has to be scanned for inaccuracies in spelling and genre assignation. Furthermore, we do not use the potentially useful *date* feature since it is often missing or inaccurate.

4.1.1 Context Event

Smart Radio is a closed domain with a finite number of playlist resources. By playing a playlist the user triggers a context event. The contents of the playlist are assumed to indicate the user's current listening preference. We term this *contextualising by instance*. In the taxonomy suggested by Lieberman, this is a "zero input" strategy in which the system uses a short term, *implicit* representation of the user's interests (Lieberman et al. 2001). Rather than extracting features from the playlist in a manner similar to Watson or Letiza, we transform the playlist representation into a *case-based* representation where the case features indicate the genre/artist mixture with in the playlist. Since the playlist is a compilation the goal is to capture the type of music mix, using the available features that would best indicate this property. Using the playlist format we are thus able to produce a much richer composite representation of the music being listened to than if we were looking solely at track descriptions.

4.1.2 Case Representation

We have two feature types associated with each track, `genre_` and `artist_`. The semantics we are trying to capture by our case representation is the composition of a playlist in terms of genre and artist, where we consider genre to be the primary feature type. The most obvious way to represent each case is to

have two features, `artist` and `genre` that contain an enumeration of the genres or artists in each feature. However, this case representation does not adequately capture the idea of a compilation of music tracks, in that it ignores the quantities of each genre/artist present in the playlist. Our case description must contain the quantities of individual genre and artists within each playlist. Furthermore, our case representation should not ignore retrieval issues. Even though we only have two features, the enumerated set of values for each feature means that similarity matching will require an exhaustive search through the case base. Since many cases will have no genres or artists in common, this is highly inefficient. Our goal is to produce a case representation that allows us to index closely matching cases, so that retrieval takes place only over the relevant portion of the case base. Finally, since one of the advantages of an instance-based representation is the ease with which explanations can be derived from retrieved cases, our case representation should be an intuitive depiction of what constitutes a compilation of music tracks.

The case representation we used in Smart Radio is illustrated in table 2. The case captures the composition of the playlist in terms of the quantity of genres and artist present. This representation allows each case to be indexed in a retrieval-efficient memory structure such as a case retrieval net which we discuss in section 5. The case mark-up demonstrated in table 2 is an example of CBML v1.0, Case Mark-up Language, which we developed to represent case data in distributed web applications (Hayes, Doyle & Cunningham 1998).

```
<case>
<casedef casename="playlist_930">
<attributes>
  <attribute name="genre_1">1</attribute>
  <attribute name="genre_11">2</attribute>
  <attribute name="genre_17">3</attribute>
  <attribute name="genre_7">4</attribute>
  <attribute name="artist_1201">1</attribute>
  ...
</attributes>
</casedef>
</case>
```

Table 2. A CBML representation of the playlist.

4.2 Feature Weights

The transformed playlist has two types of feature, *genre_* features and *artist_* features. The maximum number of features in a playlist is 20 where it is composed of 10 separate genres and 10 artists. The minimum number of features a playlist can have is two, in which case the playlist contains tracks by the same artist, and with the same genre. The currently playing playlist is used as the target for which we try and find the most similar cases available in the recommendation set. Playlist similarity is determined by matching the proportions of genre and artist contained in a playlist. When calculating playlist similarity we apply two sets of weights to theses features.

4.2.1 Feature Type Weight

The first, the *feature type weight*, is general across each query and represents the relative importance of each *type* of feature. We consider the `genre_` type more important in determining the property of playlist mix and allocate it a weight of 0.7. The `artist_` type features receive a 0.3 weighting. The reason for this is that artist features are essentially used to refine the retrieval process, boosting those playlists that match well on the `genre_` type features. This is particularly pertinent where a target playlist contains a lot of tracks by one artist. Playlists that match well on the `genre_` features are then boosted by the contribution of the `artist_` features, pushing those lists with the artist to the top of the resultset. The `artist_` features also implicitly designate the degree of mix of the target playlist. A playlist with one or two artists and one or two genres will match playlists with a similar mix while a playlist with a larger selection of genres and artists will tend to match similarly eclectic playlists. However, we recognise that the weights allocated to each feature type are based only on our view of playlist similarity. This is an inexact science based on a subjective analysis, and certainly different weight proportions per listener could be allocated were we able to easily capture each listener's outlook on playlist similarity. Stahl (2001) and Branting (2003) have proposed some techniques for determining local similarity measures, but in the context of Smart Radio it is difficult to see how these could be applied implicitly i.e. without explicitly asking the user to rate how well playlists are matched.

4.2.2 Feature Query Weight

The second weight, the *feature query weight*, is query specific and is determined by the composition of the target playlist. The *feature query weight* represents the degree of importance of each feature in determining similarity. The *feature query weight* for feature i of type t , is given as

$$wf_{t,i} = \frac{\left| f_{t,i} \right|}{\sum_{j \in t} f_j} \qquad [1]$$

where $\left| f_{t,i} \right|$ is the value for feature i of the target case. The denominator is the summation of values for features of type t. The overall weight, `ow`, for each feature is the product of the *feature type weight* and the *feature query weight*.

```
ow = feature_type_weight * feature_query_weight
```

Accordingly, the similarity weights for the features in the case in table 2 are given in table 3:

Feature	feature_type_weight * feature_query_weight	Weight
genre_1	0.7 x (1/10)	0.28
genre_11	0.7 x (2/10)	0.21
genre_17	0.7 x (3/10)	0.21
genre_7	0.3 x (4/10)	0.12
artist_1201	0.3 x (1/10)	0.03

Table 3. The table illustrates how weights are calculated on a per query basis

4.3 Similarity Metric

The similarity measure we use is given by equation 3. This measure, which is a modified form of a similarity measure known as the *weighted city block* measure (Equation 4), was chosen because it works well in matching cases where missing values occur. As we see from table 2 and table 3 the target case defines the feature (and feature weights) required in each query. Hence, on a query basis many playlist cases can be considered to have missing attribute values *with respect to the target case*. However, since each case is fully specified in its own right, many matching cases may contain genre_ features that are not relevant to the query. Even though the candidate case may closely match the target in terms of the features they have in common, the presence of irrelevant (or unsuitable) features may mean that the case is less useful than another case that only contains the target features. For this reason we apply a *similarity adjustment*, c, to each retrieved case that depends on the proportion of the *playlist* containing the genre features specified by the target (see Equation 5). This weight diminishes similarity scores that are based on a partial match with the genre_ features of the target, and preserves similarity scores that are based on full matches.

$$sim(A,B) = c\sum_{i=1}^{p} w_i \frac{|a_i - b_i|}{range} \qquad [2]$$

$$sim(A,B) = \sum_{i=1}^{p} w_i \frac{|a_i - b_i|}{range} \qquad [3]$$

$$c = \frac{|num\_tracks\_with\_target\_genre|}{|total\_number\_of\_tracks|} \qquad [4]$$

5. Integrating Context Ranking and ACF

Integrating the ACF procedure and similarity-based context ranking requires weighing up a number of factors. Burke suggests that a hybrid strategy must be a function of the characteristics of the recommender systems being combined (Burke 2002). For content and collaborative recommender systems this is largely dependent on the quantity and quality of data available. Another factor is the history of the application: is it new, in which case both techniques are untested, or is the proposed hybrid an enhancement of an already running system. For historical and logistical reasons the quantity and quality of the ACF data in the Smart Radio system is greater than the content data. Smart Radio has a greater amount of ACF data because it was originally designed and run as an ACF–based playlist recommender system (Hayes & Cunningham 2000). Content-based recommendation systems at least require a content extraction process and, in the case of knowledge-based system, they may also require a knowledge engineering process (Burke 2002). The content extraction process in Smart Radio involved mining the ID3 tags in each mp3 file which contained the `genre_` and `artist_` information. As such it was a lightweight, inexpensive process. Although, the information it yielded was not particularly rich the alternatives in the music domain are costly. While annotated music databases are available, license fees are prohibitively expensive reflecting the man-hours and knowledge required to keep up to date with the shifting music scene (AllMusic 2002). Automatically extracting *meaningful* features from music files is still a hot research topic. Researchers in a sister project to the Smart Radio project are seeking to automatically extract features from mp3 files using signal analysis which they then use to classify music tracks (Grimaldi, Cunningham, Kokaram 2003). The extraction technique, which is computationally intensive, extracts 143 low level features characterising the rhythm and frequency ranges of each music track. Although some promising results have been obtained, these features are not in a human understandable form, and a knowledge engineering task may need to be carried out on them before they can usefully be employed in a recommender system.

5.1 ACF/Content-Based Cascade architecture

The content-based strategy in Smart Radio was never designed as a stand-alone recommendation strategy. Rather it evolved through our identification of the problem of insensitivity to user-context in version 1.0 of the system. For this reason, the content-based strategy was always designed as an augmentation of the primary ACF strategy. Since one of the benefits of ACF is its 'knowledge light' approach to recommendation, our goal in designing a hybrid, content-based approach was to augment the ACF process with a similarly lightweight content-based strategy. Within the taxonomy of hybrid strategies suggested by Burke, the Smart Radio hybrid is best described as a *Cascading* system. This involves a staged process where one technique produces a candidate set which is then refined by a secondary process. Burke identifies the EntréeC system as the only other hybrid system using a *Cascading* strategy. In the EntréeC system, a content rich, knowledge-based system is the primary means of recommendation. A light ACF system is employed to decide

between tied recommendations produced by the first stage by promoting those recommendations that have been 'endorsed' by EntréeC users. The Smart Radio system, on the other hand, uses ACF as its primary recommendation strategy that is then refined by a content-based process. As a result of this, SmartRadio is an *automated* recommendation system whereas EntreeC requires the user to explicitly enter the terms of their restaurant requirements. SmartRadio is the only example in the literature of this type of automated recommendation system. The SmartRadio approach is to use the full user profile for ACF recommendations but to then refine these recommendations based on similarity to the current context. The implementation of this strategy is a type of MAC/FAC retrieval well known amongst CBR researchers (Gentner & Forbus 1991). In a novel slant on this we integrate the ACF process into this retrieval strategy.

5.2 MAC/FAC

Gentner and Forbus's MAC/FAC (Many Are Called but Few Are Chosen) retrieval technique has its origins in cognitive science where it was suggested as a model of the memory access exhibited by human beings. The technique involves a two-stage retrieval in which the MAC component provided a relatively inexpensive wide search of memory based on a surface representation of the problem. The FAC stage pruned the results from the MAC stage using a much more rigorous, examination of the structural representation of each case. In applied case based reasoning the technique has become understood as a two-stage retrieval in which a wide net search is followed by a refinement stage. Our use of the term is in this context. Our implementation of the two-stage retrieval is novel in that the first stage (MAC) is carried out by the ACF module, which returns a set of results, of which we need to decide which is the most pertinent to our user context. The second stage (FAC) involves finding matches to the context probe in the result set.

5.3 Case Memory

The Smart Radio case memory consists of the total number of playlists in the system organised as a case retrieval net with each case represented in terms of its constituent `genre_` and `artist_` features. Each playlist case can be considered to have missing features since it is impossible for a single playlist to contain all possible `genre_` and `artist_` features. As illustrated in figure 3, the case retrieval net structure will only link those cases with features in common. This ensures optimal retrieval while only traversing the relevant portions of case memory (Lenz 1999).

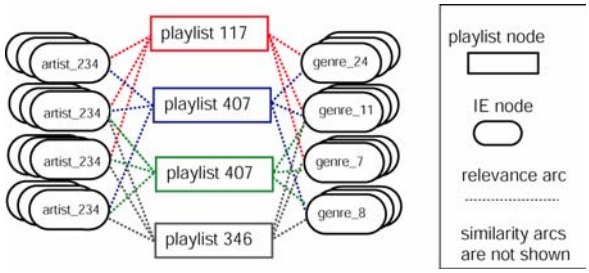

Figure 3. A schema of the playlists indexed using a case retrieval net.

As we describe in chapter 3, the ACF module also operates in a novel implementation of a case retrieval net. The output for the ACF module is a set of candidate playlists. These are the playlists the system has found using the resources of the ACF neighbourhood. The key idea at this point is that only a portion of these may be particularly relevant to the user at this time. Each retrieved playlist has a `caseIndex` which refers to the playlist case retrieval net. The set of candidate caseIndexes *primes* a subset of the case retrieval net. The context playlist is then presented as a target case. The retrieval mechanism uses a spreading activation from the target case spreads only through the primed subset of the case retrieval net. Those programmes that have highest activation after this process are those that are most similar to the target playlist. The overall activation metric is the similarity score calculated using equation 5.5. The top 5 playlists are then ranked according to their activation score.

5.4 Presenting Recommendations

The presentation strategy employed by Smart Radio is to give the user a list of ten recommended playlists per page. Smart Radio users can view the contents of any playlist with a single mouse click,. By default the top recommended playlist is displayed automatically. The further a list is from the top the less likely the user will view it (Swearingen & Sinha 2001). For this reason, the 5 most similar playlists to the context playlist out of the user's overall recommendation set are displayed at the top of the Smart Radio home page. Users quickly understand that the top 5 recommendations are particularly relevant to their listening interests at the time. After the first five playlists the recommended lists are displayed according to the predicted vote of the ACF module. As we shall see in the next section we amend this presentation strategy when we are evaluating the efficacy of our context-based recommendations.

6. Evaluation

Increasingly, there has been a demand for objective evaluation criteria for online Recommender systems. This stems from a difficulty in evaluating which recommender is better than another, and in judging which criteria to use when

making this evaluation. The most common evaluation approaches are performed *off-line* using techniques from machine learning and information retrieval such as cross validation, Leave-One-Out evaluation and measures of recall/precision. However, these techniques are not suitable for measuring the success of a recommender strategy like the content-boosted ACF that produces a ranking based on user actions at a particular time. To be sure that our new hypothesis is working we need to perform a comparative analysis of how it performs against a pure ACF strategy. We draw attention to the fact that evaluation has to measure whether real people are willing to act based on the advice of the system. Unlike the off-line analysis, this methodology plays one recommendation strategy against the other in an on-line setting and measures relative degree of success of each strategy according to how the user utilises the recommendations of either system. This framework doesn't measure absolute user satisfaction but only relative user satisfaction with one system over another. Our evaluation methodology draws upon an on-line evaluative framework for recommender systems which we have earlier defined (Hayes et al. 2002). In the interests of space we only discuss issues related to our current evaluation and refer the reader to the earlier paper for a fuller discussion. Our evaluation environment consists of a real on-line application used by a community of users, with a well-defined recommendation task using a specific user interface. The application is serviced by two competing recommendation strategies: ACF and content-boosted ACF. In order to be able to gauge a relative measure of user satisfaction with the two strategies, it is necessary to log the user interactions with respect to the recommendation engines. By comparing usage of the recommendations, it will be possible to say which strategy performed better than the other. In order to isolate the recommendation strategies we keep other aspects that might influence user satisfaction (interface, interaction model) the same. The proposed methodology can be seen as a competition between two different approaches to solving the same problem (in this case, winning user satisfaction) in which the winner is defined by how the user makes use of recommendations. We define three evaluation policies.

Presentation policy: The recommendations in Smart Radio are presented as an ordered list (figure 1). For evaluative purposes we interleave items from each strategy. Since the item presented first in a recommendation set is considered to have priority, access to this position is alternated between each recommender strategy after each playlist 'play' event.

Evaluation policy: defines how user actions will be considered evidence of preference of one algorithm over the other. In this case a preference is registered for one strategy over the other when a user plays a playlist after selecting it from the recommendation set.

Comparison metric: defines how to analyse the evaluative feedback in order to determine a winner. The simplest way is to count the number of rounds won by the competing systems. However, certain algorithms, such as collaborative filtering, may only start to perform well after sufficient data has been collected. Therefore, we also need to analyse the performance curve of each system rather than a cumulative score.

6.1 Results

The results refer to the listening data of 46 users who played a total of 606 playlists during the 65day period from 08/04/2003 until 12/06/2003. The graph in figure 4 shows the source of playlists played in the system for this period. The recommendation category was by far the most popular means of finding playlists. The category with the next highest score, the *past playlists* category, is a bit unusual since people have to have chosen a playlist from one of the other categories first before they can choose it again from their past playlists. In this case 15 of the past lists chosen were originally recommendations made within the period. We should also note that building playlists from scratch or explicitly searching for playlists should not be considered 'rival' categories to the recommendation category since an ACF based system requires users to find a proportion of new items from outside the system itself.

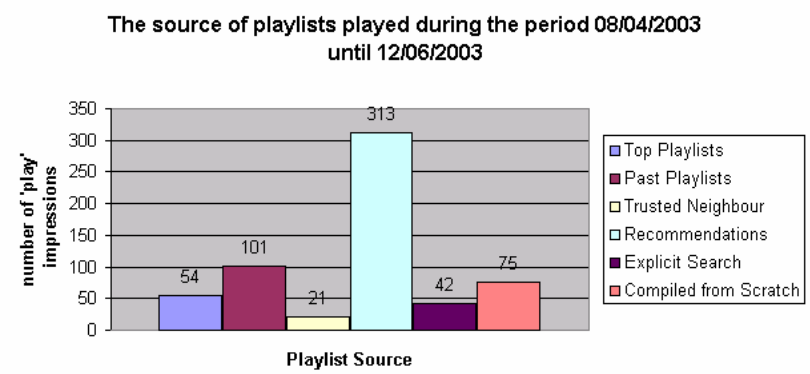

Figure 4. The source of playlist played in Smart Radio during the evaluation period.

Figure 5 illustrates the cumulative breakdown of recommendations between pure ACF recommendations and context boosted ACF for the period. From a total of 313 playlists played 190 were sourced from content boosted recommendations, while 110 came from normal ACF recommendations.

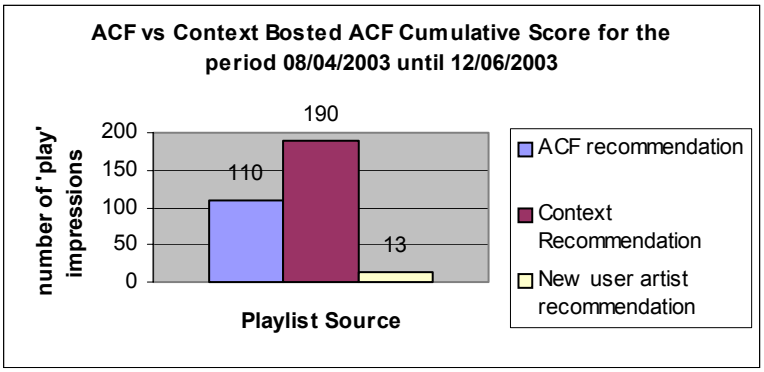

Figure 5. The cumulative scores for the ACF vs. context boosted ACF analysis

The final graph shows the proportions of ACF vs. context boosted recommendation analysed on a weekly basis for the period. The context boosted ACF continually outperformed the pure ACF recommendation strategy. In order to check that these results were consistent throughout the evaluation period we divided the period into 10 intervals of one week. The graph in figure 6 illustrates that the content-boosted ACF out performed normal ACF in each interval.

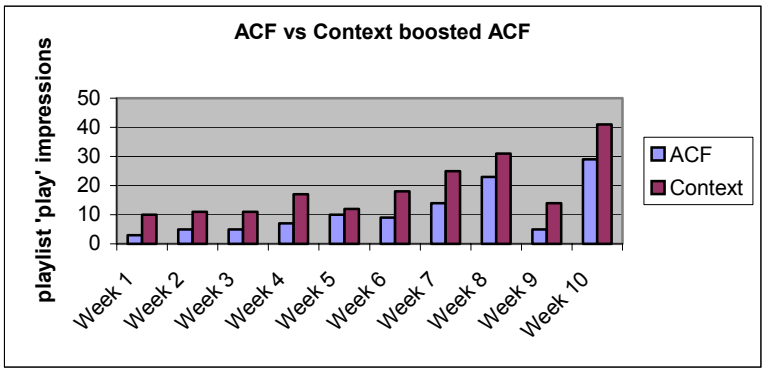

Figure 6. The graph demonstrates that the context-boosted ACF continually performed better than the pure ACF strategy throughout the evaluation period

7. Conclusion

In this paper we introduced the Smart Radio system, a community-based recommendation service where users compile and share music playlists with the aide of an automated collaborative filtering system. However, since ACF techniques are insensitive to the user's current listening preferences we have used a lightweight content-based retrieval mechanism to rank recommendations according to their relevance to the user. The principle idea is that the user's most recent interests are represented as a target case. The ACF module primes a portion

of the Case Retrieval Net memory. Using the spreading activation mechanism of the Case Retrieval Net, the ACF recommendations are then ranked according to their similarity to the target case. We evaluated our strategy with an online methodology in which both algorithms simultaneously competed. Our data would suggest that the context boosted ACF significantly out performs standard ACF.

References

1. Stahl, A. (2001). Learning Feature Weights from Case Order Feedback. Proceedings of the 4th International Conference on Case-Based Reasoning, ICCBR 2001
2. Branting, L. K. (2003). Learning Feature Weights from Customer Return-Set Selections. The Journal of Knowledge and Information Systems (KAIS). To appear, 2003.
3. Budzik, J., Hammond, K.J., Marlow, C.A., and Scheinkman, A. (1998) Anticipating information needs: Every day applications as interfaces to Internet Information sources. In proceedisngs of the 1998 Worlds Conference on the WWW, Internet ,and Intranet
4. Lieberman, H., Fry, C., and Weitzman, L., "Exploring the Web with Reconnaissance Agents," Communications of the ACM, Vol. 44, No. 8, August 2001.
5. Lieberman H, 1997. Letiza: An Agent That Assists Web Browsing" in Proceedings of the International Joint Conference on Artificial Intelligence IJCAI-95.(Montreal 1995).
6. Hayes, C., Cunningham, P. (2001) SmartRadio–community based music radio; Knowledge Based Systems, special issue ES2000, Volume 14, Issue3-4, June 2001, Elsevier
7. Shardanand, U., and Mayes, P., (1995) Social Information Filtering: Algorithms for Automating 'Word of Mouth', in Proceedings of CHI95, 210-217.
8. Schafer, J.B., Konstan, J.A., and Riedl, J. 1999. Recommender Systems in E-Commerce. In ACM Conference on Electronic Commerce (EC-99), pages 158-166.
9. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J. An Open Architecture for Collaborative Filtering of Netnews. pages 175--186. ACM Conference on Computer Supported Co-operative Work, 1994.
10. Balabanovic, M., Shoham, Y. (1995) Learning Information Retrieval Agents: Experiments with Automated Web Browsing, in AAAI Spring Symposium on Information Gathering, Stanford, CA, March 1995
11. Schlit B. et al, Context-Aware Computing Applications, IEEE Wokshop on Mobile Computing Systems and Applications 1994
12. Burke, R. (2002) Hybrid Recommender Systems: Surveys and Experiments in User Modeling and User-Adapted Interaction 12(4): 331-370; Nov 2002. Kluwer press.
13. Allmusic (2002). E-mail correspondence on licensing arrangements for the access to the allmusic.com database
14. Grimaldi, M., Cunningham, P., Kokaram, A. (2003) An Evaluation of Alternative Feature Selection Strategies and Ensemble Techniques for Classifying Music. Technical report TCD-CS-2003-21, Computer Science Department, Trinity College Dublin.
15. 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
16. Lenz, M., (1999) Case Retrieval Nets as a model for building flexible information systems. PhD dissertation, Humboldt University, Berlin. Faculty of Mathematics and Natural Sciences.
17. Swearingen, K., Sinha, R., (2001) Beyond Algorithms: An HCI Perspective on Recommender Systems, ACM SIGIR Workshop on Recommender Systems.
18. Hayes, C., Massa, P., Avesani, P., Cunningham, P., (2002) . An on-line evaluation framework for recommender systems in the proceedings of the IWorkshop on Recommendation and Personalization Systems, AH 2002, Malaga, Spain, 2002. Springer Verlag.