Re-using Implicit Knowledge in Short-term Information Profiles for Context-sensitive Tasks

Conor Hayes, Paolo Avesani, Emiliano Baldo[1] & Pádraig Cunningham[2]

[1] ITC-IRST, Via Sommarive 18, 38050 Povo, Trento, Italy
{hayes, avesani, baldo}@itc.it
[2] Department of Computer Science, Trinity College Dublin
Padraig.Cunningham@cs.tcd.ie

Abstract. Typically, case-based recommender systems recommend single items to the on-line customer. In this paper we introduce the idea of recommending a user-defined *collection* of items where the user has implicitly encoded the relationships between the items. Automated collaborative filtering (ACF), a so-called 'contentless' technique, has been widely used as a recommendation strategy for music items. However, its reliance on a global model of the user's interests makes it unsuited to catering for the user's local interests. We consider the context-sensitive task of building a compilation, a user-defined collection of music tracks. In our analysis, a collection is a case that captures a specific short-term information/music need. In an offline evaluation, we demonstrate how a case-completion strategy that uses short-term representations is significantly more effective than the ACF technique. We then consider the problem of recommending a compilation according to the user's most recent listening preferences. Using a novel on-line evaluation where two algorithms compete for the user's attention, we demonstrate how a knowledge-light case-based reasoning strategy successfully addresses this problem.

1 Introduction

There have been many research and application based initiatives promoting case-based reasoning (CBR) as a suitable recommender methodology for on-line services [32,7]. Increasingly CBR has been employed to provide solutions to customers purchasing *configurable* products such as personal computers, holidays and electronic equipment. Meeting the customer's requirements for a configurable entity requires some additional knowledge encoding the constraints and dependencies between components. This type of *expert* knowledge can be deployed during the adaptation [29] or retrieval process [25].

In this paper we introduce the concept of recommending a *collection* of items using case-based techniques. A collection is a set of items that have been assembled by a user according to a particular idea or motivation. In this sense it has much in common with a configurable entity. However, it differs in that the relationships between component parts cannot be described *a priori* in a formal way. Rather, they are *implicitly* encoded by the user during his/her construction of the collection. The goal

of this work is to make use of these implicitly encoded *rules of thumb* when providing advice to other users.

Our research is based upon experiments conducted on data from the Smart Radio system, an online streaming music system which allowed users to build collections of music (compilations) which could be recommended to users with similar tastes [17, 16, 15]. Although the techniques described in this paper can be applied more widely, the music domain is a classic example in which an acute knowledge elicitation bottle neck applies [9,11], making it very difficult to extract the expert rules that encode the relationships between music items. However, with the arrival of new on-line music services, consumers are faced with the familiar problem of *information overload* often described for textual material [12,27]. As such, automated collaborative filtering (ACF), a so-called 'contentless' approach to recommendation and personalisation, has dominated in applications in the music domain [16,30,20]. One serious drawback with ACF is that it is not able to make recommendations that are sensitive to the local interests or activities of the user [21]. This is because an ACF user profile is simply a vector of item Ids containing no type, ordering or session information that could be used to make recommendations on a particular subject area or for a particular task. Generally, the sparsity of the available data requires us to use the accumulated ratings of a user to make sound correlations with other users. However, the resulting recommendation set will reflect the user's accrued interests rather than the requirements of a localised task.

In this paper we consider two context-sensitive tasks in this domain that cannot be satisfactorily performed using the ACF algorithm.

1. Providing advice to the user when he/she is building a compilation.
2. Recommending a new compilation based on the user's current listening preference.

We will demonstrate how simple CBR-based enhancements allow us to provide accurate recommendations while still obeying the domain constraint of a knowledge-light approach. A key idea here is that we do not use long-term user profiles to make recommendations. Instead we consider each compilation to represent a short-term, context-specific need. We consider the compilation-building task as an incremental *case completion* exercise. As each track is added to a target compilation, similar compilations are retrieved and the tracks extracted from these compilations are offered to the user to complete the partial compilation. In order to counter the problem of calculating similarity between non-overlapping compilations, we use the *agave* algorithm to represent our compilations in terms of their probabilistic 'relatedness' to other tracks [1,2]. In our offline evaluation of the compilation completion technique, we demonstrate that using short-term profiles significantly outperformed the standard ACF algorithm. This would suggest that the implicit knowledge in short-term representations can be leveraged to provide advice in task-specific contexts.

We then address the problem of automatically providing a suitable follow-up compilation based on the user's current listening preferences. Using a MAC/FAC [14] approach, compilations retrieved by the ACF algorithm are re-ranked using a knowledge-light, case-based process. In contrast to the evaluation of compilation completion, we demonstrate how an *on-line* evaluation can give a true indication of user satisfaction with one algorithm over another. In this contest between algorithms,

we find that users are significantly more response to context-boosted ACF over standard ACF.

Section 2 briefly describes the Smart Radio system operation, and introduces the idea of a compilation, a user-defined collection of music tracks. We also introduce some of the principles of ACF. In section 3 we describe our solution to the compilation-building task using a case completion strategy, and in section 4 we introduce the idea of a *context* and the strategy we use to further refine recommendations made by the ACF engine. Section 5 presents an *offline* evaluation of the compilation completion technique. By contrast, section 6 introduces an *online* evaluation of the context-boosted ACF technique. Finally, we discuss our conclusions and future work in section 7.

2 Music Recommendation

Although a late starter, the online retail of music has grown rapidly over the past two years. With record companies increasingly making their back catalogues of music available on line, consumers are presented with an information overload problem. This problem is exacerbated by the fact that, unlike documents, which can be rapidly downloaded and scanned for relevance, a music file has a longer download time and must be listened to in real time to determine its relevance.

Smart Radio is a web-based client-server application that allows users to build compilations of music that can be streamed to the desktop [17,16,15]. The idea behind Smart Radio is to encourage the sharing of music programmes using automated recommendation techniques. The unit of recommendation in Smart Radio is the *compilation,* a collection of music tracks assembled on the fly by one listener and recommended to other like-minded listeners.

Fig. 1. A screen shot of the Smart Radio recommendation screen

In contrast to text-based information retrieval where a page of text can be automatically parsed into a vector of words and indexed for retrieval, there are *no* fundamental units of meaning equivalent to words which can be applied to music retrieval [13,9,11]. Furthermore, experts in the area of music come from disparate disciplines such as musicology, music cognition, signal processing and music

librarianship, and there is great difficulty in integrating the knowledge produced in these fields [11]. While music information retrieval systems may appear to be decades behind text retrieval [9], it is clear that, with the increasing amount of digital music online, there is a pressing need for music personalisation/recommendation techniques.

Thus, music recommender systems such as Smart Radio generally rely upon techniques such as ACF where explicit content mark-up is not required. The key idea in ACF is that users can be clustered together based on their usage patterns. Recommendations can be made to a target user based on the accumulated data in neighbouring user profiles. As similarity between user profiles is calculated based on the intersection of the item ids between profiles and not on content description, ACF allows recommendations to be made in domains like music where there is a *knowledge-elicitation* bottleneck. A second strength of ACF is that it can make recommendations that would otherwise escape content-based recommender strategies. This is because it relies upon *implicit knowledge* expressed in user preferences that may capture the subtle relationship between items that would otherwise escape a content-based system [10]. It is this type of *implicit* knowledge that we wish to explore in this paper.

Recent work in the CBR community has drawn a parallel between ACF and CBR as case completion [18,2,24]. The observation in this work is that the typical ACF mechanism of incrementally offering recommendations based on user feedback is like the case completion process in CBR. In this analogy, the ACF user profile is the incomplete case and recommended items are proposed solutions for case completion. Despite this, ACF suffers from some weaknesses not apparent in CBR systems. Its key strength in recommending surreptitious items is also a prominent weakness. Whereas the CBR case has typically been viewed as capturing a single problem/solution episode, a single ACF user case may capture several heterogeneous episodes reflecting the various interests of the user over time. The ACF recommender does not take into account the nature of the content it recommends, and thus, while the items it recommends may be suitable in relation to the overall interests of the user, they may be entirely unsuitable for the current search task to hand. In the next section, we describe a solution to this problem by using short-term, task-oriented profiles.

3 Compilation Building as Case Completion

A compilation is a user-defined collection of music, very often made up of a mixture of tracks by different artists, but not necessarily so. We view a compilation as a case that captures a particular short-term music/information requirement. Apart from its component tracks, it also implicitly contains the knowledge and work required to assemble the collection. Our hypothesis is that each compilation is built according to an implicitly articulated guiding principle. Thus, each compilation case inherently contains information about the relatedness between component tracks. Indeed, it has been demonstrated that a similar type of 'relatedness' information, useful for accurate recommendation, can be mined from lists of favourite artists posted by music fans on the internet [10].

In Smart Radio, a compilation consists of a collection of 10 tracks which is assembled for *immediate* delivery using streaming protocols. Thus, the user must choose the composition of the compilation with care because once the compilation has started to play, its composition cannot be modified. For this reason, we consider that the compilations in Smart Radio should contain implicit knowledge as to what tracks 'fit' together. In the next subsection, we describe how we allow users to reap the benefit of the compilation-building expertise of previous users.

Fig. 2. A screen shot of a compilation advisor system

3.1 Providing Compilation Completion Advice

Building a compilation is a context-sensitive task where tracks are selected according to a particular theme or idea of the user. The key to this approach is to realise that a short-term profile can capture a single problem-solving episode that is better able to provide context-sensitive recommendations. Thus, instead of using user profiles to make recommendations we choose to tap the specific knowledge in other compilations. We draw our methodology in part from the interactive case completion framework of the NaCoDae system [3]. As the user adds tracks to a compilation, the compilation adviser retrieves similar compilations and offers the user a choice of tracks or full compilations to complete his/her compilation. Figure 2 illustrates a screen shot for a compilation adviser we have implemented in the music domain [2]. The user, Conchuir, has added two tracks to his compilation. In response, the adviser immediately presents a list of tracks that would suitably complete this compilation. The user can choose these tracks or may select full compilation solutions from the compilations tab.

3.2 Data Sparsity

Using a compilation as a short-term profile is problematic in that similarity can only be measured on items shared in common between compilation profiles. A compilation is made up of 10 out of a possible 2148 tracks. Clearly, many compilations cannot be compared because they do not have any tracks in common. In contrast, a typical user profile would contain tracks from many sessions and thus have a better chance of

intersecting with other profiles. We address this problem using the *agave* algorithm to reduce the sparsity of the compilation data set [1,2]. Using *agave*, each compilation is transformed to a less sparse representation that reflects the degree of relatedness of each compilation to each of the other 2148 tracks available. This value is called the *mu* value of the track with respect to a particular compilation. For each compilation, *t*, the *mu* feature value for track *u* is the summation of the conditional probability that track *u* co-occurs with compilation track, t_i. The *mu* is normalised by *n* the number of non-zero values for $P(t_i/u)$. *n* is thus the number of values for which there is not a zero probability of t_i and *u* co-occurring.

$$mu = \frac{\sum P(t_i / u)}{n} \quad \textbf{(1)}$$

Thus, for each compilation, we can produce a *mu* value for each track available, transforming the sparse compilation representation into a vector of *mu* values. Intuitively, no calculation of *mu* is possible for track *u* if it is not co-frequent at least once with t_i in any compilation in the compilation data set. Avesani & Aguzzoli have demonstrated that *agave* performs better than singular value decomposition (SVD), another technique for reducing sparsity in ACF data sets [2]. Whereas SVD operates by collapsing the dimensions of the data set, *agave* has the advantage of preserving the original dimensions. In our evaluation we also use the *PSim* approach, which also addresses the similarity coverage problem in sparse data sets [24]. However, an analysis of this technique would suggest that it relies on exactly the same principles as *agave*: the conditional probabilities of co-frequent pairs in the data set.

4 Recommending Compilations in Context

Whereas in section 3 we described how we used the implicit compilation-building knowledge to help the user build a compilation, we now turn our attention to recommending a full compilation. Many of the same issues still apply. Using a typical ACF strategy, recommended compilations may not suit the user's current compilation preferences. Our goal is to make recommendations that are appropriate within the user's listening context.

In the field of user modelling, the objective in isolating context information is that tasks being undertaken by the user may be anticipated and a portion of the work carried out automatically in advance. Applications such as Watson [6] and Letizia [22,23] which monitor the user's behaviour and attempt to retrieve or predict relevant information have been termed 'reconnaissance aides'. In both Watson and Letizia the context is represented by a content-based analysis of the topics currently of interest to the user. 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 it is a "zero input" strategy, i.e. the user does not need to explicitly describe his/her goals, prior to working [23]

Our goal is to enhance the ACF technique so that compilations based on the user's current context are promoted but requiring zero input from the user. Our approach is to use a MAC/FAC influenced methodology in which an 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 in Figure 3.

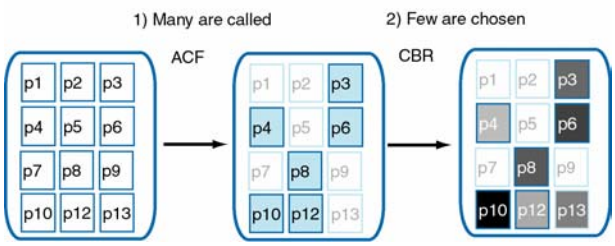

Fig. 3. The two-stage strategy for providing context-sensitive recommendations

Unlike the examples of the *reconnaissance* aides described above, which used information retrieval analyses to build a short-term user profile, the Smart Radio domain suffers from a deficit of content descriptions. Therefore, the solution is to use a lightweight case-based representation of each compilation using some freely available meta-data. The content descriptors we use are the *genre* and *artist* tags found in a few bytes of information at the end of the mp3 file. Although the information this inexpensive process yielded was not particularly rich, the alternatives in the music domain are expensive. We transform the compilation representation into a *case-based* representation where the case features indicate the *genre/artist* mixture within the compilation. As the goal is to capture the type of music mix, using the available features that would best indicate this property. We have two feature types associated with each track, `genre_` and `artist_`. The case representation we used in Smart Radio is illustrated in Table 2.

Table 1. A case captures the composition of the compilation in terms of the quantity of genres and artists present. For reasons of space only one artist feature is shown.

Feat. type	Feature	Value
genre_	Jazz	1
genre_	Blues	2
genre_	Folk	3
genre_	Country	4
artist_	John Coltrane	1

By playing a compilation the user triggers a context event. The contents of the compilation are assumed to indicate the user's current listening preference. We term this *contextualising by instance.* The transformed compilation has two types of feature, `genre_` features and `artist_` features. The currently playing compilation is used as the target for which we try and find the most similar cases available in the ACF recommendation set. Compilation similarity is determined by matching the proportions of genre and artist contained in a compilation. In section 6, we present an online evaluation of this technique where we test user response to recommendations presented from the context-boosted ACF strategy and an ACF strategy.

4.1 Integrating Context Ranking and ACF

The content-based strategy in Smart Radio 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. Within the taxonomy of hybrid strategies suggested by Burke, the Smart Radio hybrid is best described as a *Cascading* system [8]. Unlike the *EntreeC* system, another type of Cascading hybrid, the Smart Radio system uses ACF as its primary recommendation strategy and the content-based ranking as a supplemental process. A complete description of the integrated ACF–CBR approach we adopt is beyond the scope of this paper. Readers are directed to Ref. 15 for a more in-depth discussion of the similarity techniques and the architecture we use.

5 Off-line Evaluation of Case Completion

As we described in section 3, Cocoa compilations are built according to the 'expert knowledge' of users. In this section we describe an offline evaluation in which we had the following objectives:

1. To demonstrate that short-term information profiles are more successful than typical long-term profiles for a context-sensitive task such as compilation completion
2. To evaluate whether completion information should be based on individually retrieved compilations or an aggregation of tracks from the *k*-nearest neighbours.

5.1 Compilation Completion

Our evaluation strategy involved simulating a case completion process whereby we measured recall at different stages of case completion. The recall measure represents the probability that a relevant item will be retrieved. Each compilation in the case base is a unique, user-defined collection of music. A *leave-one-out* approach was used whereby we removed a percentage of tracks for each compilation. By retrieving a set of *k* nearest compilations we then attempted to predict the missing tracks. In order to simulate performance at difference levels of completion, the missing tracks were removed in increments of 10%. In calculating recall at each percentage of the partially completed compilation, the relevant set refers to the set of items removed. The algorithms we used are described below.

TopN: This technique was used as a baseline approach. We recommend the *N* most frequent tracks in the data set

Overlap_Userbased_knn_topN: This is the standard user-based ACF algorithm. We represent the data in the training set as a set of long-term user profiles containing tracks from the compilations that the user has downloaded in the past. Using the overlap method, recommendations are made by firstly retrieving the best matching user profiles (*k*nn) for the target compilation and then choosing the most frequently occurring items in the retrieved profiles (topN) [28]. However, as each user profile

has a binary representation in terms of tracks, similarity between the target and candidate profiles is based on the amount of overlapping tracks.

Overlap_Comp-based_knn_topN: We then retrieve compilations rather than user profiles using the *overlap* method as before. Again, track recommendations are made by choosing the most frequently occurring items in the retrieved profiles.

The next three approaches use *agave* sparsity reduction. We make recommendations based on the first *k* compilations retrieved.

Agave_P_knn: We use the *agave* sparsity reduction technique. The similarity metric is the Pearson coefficient. Recommended items are presented in the order they occur in the *k* ranked compilations.

Agave_P_knn_rmv_dupl: This is the same as Agave_P_Knn except that items already in the target compilations and any lower ranking duplicate items are removed.

Agave_LS_knn_topN: We use the *agave* sparsity reduction technique. The similarity metric is based on the Least Squares metric used by Shardanand and Maes [30]. Recommended items are ranked according to their frequency in the *k* compilations.

PSim approach: This is an implementation of the approach described in Ref. 24. The *PSim* approach and *agave* techniques are based respectively on *Isim* and *mu* measures. It can easily be shown that *mu* and *ISim* are equivalent.

5.2 Evaluation methodology

The Smart Radio data set contains 803 compilations built by listeners to the Smart Radio system from a corpus of 2148 tracks. Each compilation has 10 tracks.

Each compilation in the data set is evaluated using the *leave-one-out* methodology. When we use the *agave* or *PSim* approach we recalculate the *mu* or *ISim* scores using the data set minus the compilation being tested. For each compilation test, recall is measured at incremental stages of completion. For example, in the first test we remove 90% of the compilation. The remaining 10% is used as the target and the 90% we removed acts as the relevant set with which we can calculate the recall score for the retrieved tracks. We continue to test in increments of 10% until we finally evaluate recall when 90% of the compilation is present and 10% acts as the relevant set. In each test, the retrieval size is set at 10 compilations. In measuring recall, we consider the ranked list of tracks produced by each algorithm. Tracks already found in the target compilation or duplicates of tracks already ranked higher in the retrieval list are considered non-relevant items for the purpose of calculating recall. Thus, algorithms which produce duplicates or include tracks already found in the target compilation are penalised. For every retrieval algorithm we set $k = 10$. However, the *Agave_P_knn* variants only used the track data in the first or second compilation.

5.3 Results

Figure 4 illustrates the Recall graph for case completion where the x axis represents the percentage of the compilation used as the target compilation. Clearly, the *topN* and *user-based* approach perform very poorly when faced with a context-specific task. The *comp-based* approach, which utilises short-term profiles in the form of other

compilations, performs significantly better even though the similarity is based only on compilation overlap.

However, the algorithms which use the short-term profiles and the *agave* (or PSim) sparsity reduction techniques perform best overall. There is no significant winner amongst the various permutations we tried. Some performed a little better at early stages of the case completion process and others at late stages.

The significant difference between the performance of the user-based approach and the approaches based on compilation retrieval would seem to be due to the loss of context information in the user profiles.

One of our objectives was to test whether presenting compilations in the order they are ranked by the similarity metric is an adequate recommendation strategy. Our hypothesis is that the first 1 or 2 ranked compilations are likely to contain sufficient track information to complete the test compilation. In fact, *agave_P_knn* and *agave_P_knn_rmv_dupl* perform very well indeed. These algorithms represent the view the user would have when choosing the 'compilations' tab in Figure 2. All the other algorithms aggregate the track data from the *k* nearest compilations, ranking them by frequency, for example. This is equivalent to the view in the 'tracks' tab of Figure 2. Our evaluation would suggest that the knowledge contained in the first two top-ranking compilations is strong enough to compete with the aggregated data from *k* compilations.

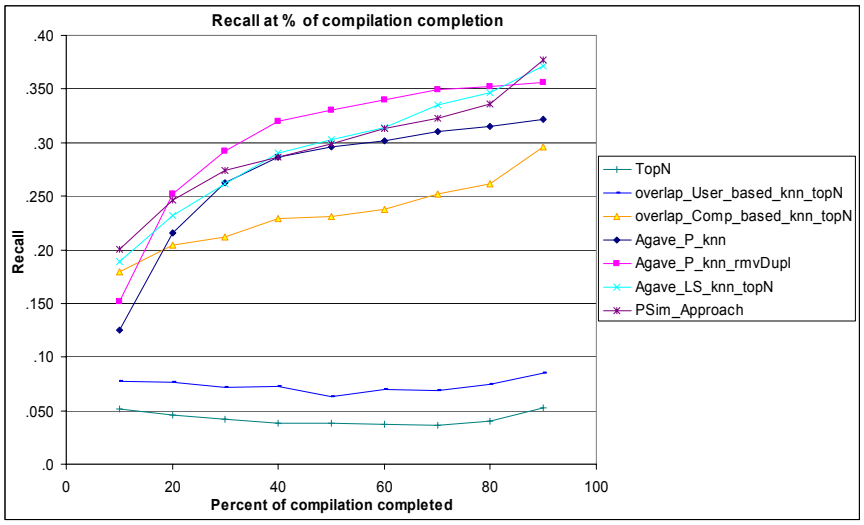

Fig. 4. Recall graph for compilation completion

6 An *Online* Evaluation of Context-boosted ACF vs. ACF

The evaluation in section 5 is an example of an *off-line* evaluation of a recommender strategy which is typically based on techniques in machine learning and information

retrieval [5]. However, it has been regularly observed that off-line evaluations of recommender systems have a number of shortcomings [31,19,21]. For example, it is not at all clear whether users are sensitive to slight improvements in prediction error of one algorithm over another. Secondly, an algorithm can only be evaluated on predictions it makes on items that have been observed by the user, which may be only a fraction of the overall items in the domain. Thus, in an offline evaluation, there is no way of measuring 'true recall' because we are unable to measure the potential relevance of items that have not been rated by the user.

This problem is particularly apparent when evaluating the success of a recommender strategy like the content-boosted ACF where we need to analyse the correctness of the ranking produced in response to a context event. It was not clear how we might perform this in an off-line setting. Therefore, to test our hypothesis we performed a comparative analysis of how the algorithm performs in an *online* setting. The key idea of online evaluation is that we 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 a live system and measures the *relative* degree of success of each strategy according to whether the user utilises the recommendations of either system. A more detailed discussion of our on-line evaluation framework for recommender systems is presented in Ref. 19.

6.1 Evaluation Environment

The evaluation environment was the Smart Radio system – a live, on-line application used by a community of users, with a well defined recommendation task using a specific user interface. The application was serviced by two competing recommendation strategies: ACF and context-boosted ACF. In order to be able to gauge a relative measure of user satisfaction with the two strategies, we logged the user interactions with respect to the recommendations made by either strategy. Other aspects of the recommendation process that might have influenced user satisfaction were kept the same (interface, interaction model). 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 this regard, we define three evaluation policies.

Presentation policy: The recommended compilations in Smart Radio were presented as a ranked list. For evaluative purposes, we *interleaved* recommendations from each strategy. As a user is most likely to inspect the top-ranked compilation in the recommendation set, this position is alternated between each recommender strategy after each compilation 'play' event.

Evaluation policy: defines how user actions can be interpreted to express a preference for one algorithm over the other. In this evaluation, a preference was registered for one strategy when a user inspected and then played a compilation from his/her recommendation set.

Comparison policy: defines how to analyse the evaluation data in order to determine a winner. Obviously, the simplest way is to count the number of rounds won by the competing systems. However, certain algorithms, such as ACF, may only start to perform well after sufficient data has been collected. Therefore, we analyse the performance of each system over time. As individual users may have different

degrees of interaction with the system, we also make a comparative analysis of different types of users.

6.2 Results

The results refer to the listening data of 58 users who played a total of 1012 compilations during the 101-day period from 08/04/2003 until 17/07/2003. Table 2 gives the breakdown of the sources for compilations played in the system for this period. The recommendation category was by far the most popular means of finding compilations. We should also note that building compilations from scratch or explicitly searching for compilations should not be considered 'rival' categories to the recommendation category given that an ACF-based system requires users to find a proportion of new items from outside the recommendation system itself.

Cumulative Score: Table 3 gives the cumulative breakdown between ACF and context-boosted ACF recommendations for the period. From a total of 504 recommended compilations played, 311 were sourced from content-boosted recommendations, while 177 came from normal ACF recommendations. 16 came from bootstrap recommendations which we haven't discussed here.

Interval-Based Evaluation: In order to check that these results were consistent throughout the evaluation period, we divided the period into 15 intervals of one week. Figure 8 shows the proportions of ACF to context-boosted recommendations analysed on a weekly basis for the period. We can see that the context-boosted ACF continually outperformed the pure ACF recommendation strategy. We have tested these results using a paired *t*-test and found them to be statistically significant within a confidence level of 99%.

User-based Evaluation: An analysis of our users' behaviour demonstrated considerable variance. During the evaluation period we had users who used the system several times a week, sometimes for hours every day, as well as other users who used the system much less frequently. In order to check that the performance of our recommender holds for different degrees of usage, we split the dataset according to the number of compilations each user listened to. There are 10 categories in which users may fall, representing different degrees of usage of the system. Figure 9 illustrates the comparative success of the two strategies in each usage range.

Whilst ACF is marginally greater in two intervals, if we use a paired *t*-test on the individual user recommendation data we find that the hypothesis, ACF $\leq$ context-boosted ACF once again holds with a confidence level of 95%. However, Figure 3 would suggest that the preference for context-boosted ACF is more pronounced among regular users of the system. Light users simply might not have used the system enough to have formed a preference for either recommendation strategy. Heavier users, on the other hand, have a much greater chance to explore the facilities of the system and implicitly express preferences for one strategy over another through regular use.

Table 2. Source of compilations played from 24:00 08/04/2003 until 24:00 17/07/2003

Source	Number	Percentage
Top Compilations	87	8
Past Compilations	194	19
Trusted Neighbour	23	2
Recommendations	504	50
Explicit Search	94	9
Compiled from Scratch	110	11

Table 3. The cumulative scores for the ACF vs. context-boosted ACF analysis

Algorithm name	Number of 'play' impressions	Percentage
Standard ACF	177	35
Context-boosted ACF	311	62

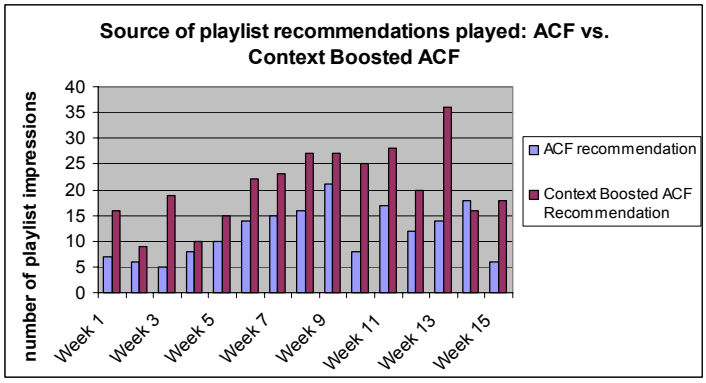

Fig. 5. ACF vs. context-boosted ACF over 15 weekly intervals

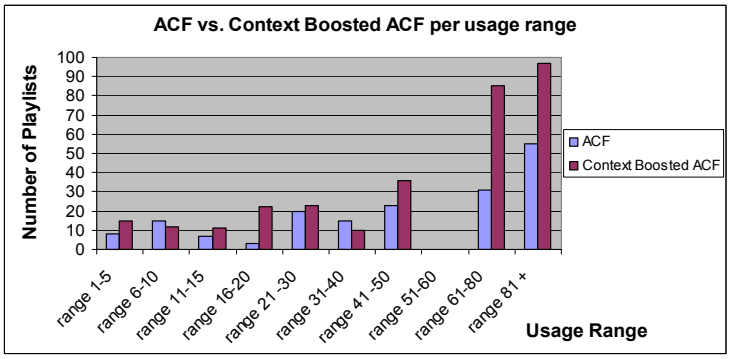

Fig. 6. A user-based analysis of the evaluation

7 Conclusions

In this paper we demonstrate the importance of considering the context of the online user's interests or tasks. In the domain of music, however, there is great difficulty in extracting content or knowledge with which to model user profiles. Conventionally, the ACF technique is used. However, ACF makes recommendations based on a global model of the user's interests. We show how short-term profiles in the form of collections of music are much more successful in providing advice in the compilation-building exercise. The key observation we make is that such short-term collections contain implicit knowledge as to the relatedness of their component tracks. However, we note that typical off-line approaches are limited to evaluating algorithmic performance on items the user has rated in the past. In our second evaluation we demonstrate how an online test gives evidence of user satisfaction with one strategy over another. In particular, we show user preference for a context-enhanced ACF algorithm over a standard ACF algorithm.

8 References

1. Aguzzoli, S., Avesani, P., Massa, P. Compositional CBR via collaborative filtering. In ICCBR '01 Workshop on CBR in Electronic Commerce, Vancouver, Canada, 2001.
2. Aguzzoli, S., Avesani, P., Massa, P. Collaborative case-based recommender systems. In ECCBR 2002, Aberdeen, Scotland, Springer Verlag, 2002.
3. Aha, D. W., Maney, T., Breslow, L. Supporting dialogue inferencing in conversational case-based reasoning. EWCBR 1998, Dublin, Ireland, pp. 262–273, 1998.
4. Bauer, T., Leake, D.B. WordSieve: a method for real-time context extraction. In Proceedings of the Third International and Interdisciplinary Conference, Context 2001, Springer, Berlin, 2001.
5. Breese, J.S., Heckerman, D., Kadie, C. Empirical analysis of predictive algorithms for collaborative filtering. In Proceedings of the 14th Annual Conference on Uncertainty in Artificial Intelligence, pp. 43–52, July 1998.
6. Budzik, J., Hammond, K. J. User interactions with everyday applications as context for just-in-time information access. In Proc. of the 2000 International Conference on Intelligent User Interfaces, (New Orleans, Louisiana, USA), ACM Press, 2000.
7. Burke, R., (ed). Proceedings of the workshop on Case-based Reasoning in Electronic Commerce. ICCBR, Vancouver, BC. 2001
8. Burke, R., Hybrid Recommender Systems: Surveys and Experiments in User Modelling and User-Adapted Interaction 12(4): 331-370; Kluwer press, Nov 2002..
9. Byrd, D., Crawford, T. Problems of music information retrieval in the real world, Information Processing and Management: an International Journal, v.38 n.2, 2002
10. Cohen, W., Fan, W. Web-collaborative filtering: Recommending music by crawling the web. In Proceedings of the Ninth International World Wide Web Conference, 2000.
11. Downie, J. Stephen. Music information retrieval (Chapter 7), In: Cronin, Blaise (Hg.) Annual Review of Information Science and Technology 37: Information Today Books, pp.295-340, 2003
12. Foltz, P.W., Dumais, S.T. Personalized information delivery: An analysis of information filtering methods. Communications of the ACM 35(12), 51–60, 1992
13. Foote, J., an overview of Audio information retrieval. Multimedia Systems 7: 2–10 Springer Verlag, 1999.

14. Gentner, D., Forbus, K. D., MAC/FAC: A model of similarity based access and mapping. In Proc. of the 13th Annual Conference of the Cognitive Science Society. Erlbaum
15. Hayes, C., Cunningham, P. Context Boosting Collaborative Recommendations. In the Journal of Knowledge Based Systems, Volume 17, Issue 5-6, July 2004, Elsevier, 2004
16. Hayes, C., Cunningham, P., Clerkin, P., Grimaldi, M. Programme-Driven Music Radio. In the proc. of ECAI 2002, Lyons France ed.: Frank van Harmelen, IOS Press, 2002
17. Hayes, C., Cunningham, P., SmartRadio–community based music radio; Knowledge Based Systems, special issue ES2000, Volume 14, Issue3-4, , Elsevier, 2001
18. Hayes, C., Cunningham, P., Smyth, B. A case-based reasoning view of automated collaborative filtering, in: Aha, D.W., Watson, I. (Eds.), Proc. of 4th International Conference on Case-Based Reasoning, LNAI 2080. Springer Verlag, pp. 234–248, 2001
19. Hayes, C., Massa, P., Avesani, P., Cunningham, P., 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.
20. Hayes, C., Smart Radio: Building Community Based Radio. PhD thesis. Department of Computer Science. Trinity College Dublin, 2004
21. Herlocker, J. L., Konstan, J. A., Terveen, L. G., and Riedl, J. T. Evaluating Collaborative Filtering Recommender Systems. In Proceedings of the ACM Transactions on Information Systems, vol. 22, no. 1, pp. 5-53, 2004
22. Lieberman H, Letiza: An Agent That Assists Web Browsing" in Proceedings of the International Joint Conference on Artificial Intelligence IJCAI-95.(Montreal 1995).
23. Lieberman, H., Fry, C., and Weitzman, L., Exploring the Web with Reconnaissance Agents," Communications of the ACM, Vol. 44, No. 8, August 2001.
24. O'Sullivan, D, Wilson, D.C., Smyth, B. Improving Case-Based Recommendation: A Collaborative Filtering Approach." In Proceedings of the Sixth European Conference on Case Based Reasoning. LNAI 2416 pp. 278-291, 2002
25. Prasad, M.V. N., Plaza, E., Corporate Memories as Distributed Case Libraries. In Proceedings of the Corporate Memory and Enterprise Modelling Track in the 10th Knowledge Acquisition Workshop. Banff, Canada, 1996.
26. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J. An Open Architecture for Collaborative Filtering of Netnews. pp. 175-186. ACM Conference on Computer Supported Co-operative Work, 1994.
27. Resnick, P., Varian, H. R. Recommender Systems. Communications of the ACM 40(3), 56–58, 1997
28. Sarwar, B., Karypis, G., Konstan, J., Riedl, J. Analysis of recommendation algorithms for e-commerce, in: Proceedings of ACM E-Commerce, 2000
29. Schmitt,S., Bergmann, R. Applying Case-Based Reasoning Technology for Product Selection and Customization in Electronic Commerce Environments. In Proc. of 12th International Bled Electronic Commerce Conference, Bled, Slovenia, June 7 - 9, 1999
30. Shardanand, U., and Mayes, P., Social Information Filtering: Algorithms for Automating 'Word of Mouth', in Proceedings of CHI95, 210-217, 1995.
31. Swearingen, K., Sinha, R., Beyond Algorithms: An HCI Perspective on Recommender Systems, ACM SIGIR Workshop on Recommender Systems, 2001.
32. Wilke, W., Lenz, M., Wess, S. Case-Based Reasoning for Electronic Commerce. In: Lenz et al. (Eds.): Case-Based Reasoning Technology from Foundations to Applications, Springer, 1998.