

Smart Radio:
Building Community-Based
Internet Music Radio

Conor Hayes

A thesis submitted to the
University of Dublin, Trinity College
for the degree of
Doctor of Philosophy

October 2003

Declaration

This thesis has not been submitted as an exercise for a degree at any other University. Except where otherwise stated, the work described herein has been carried out by the author alone. This thesis may be borrowed or copied upon request with the permission of the Librarian, University of Dublin, Trinity College.

The copyright belongs jointly to the University of Dublin and Conor Hayes.

Signed:

Conor Hayes

October 2003

Acknowledgements

I would like to express my gratitude to my supervisor, Dr. Pádraig Cunningham, for his advice and friendship over the past five years. Pádraig's dedication to his students is second to none, and this was demonstrated to me in the speed and thoroughness with which he proofed the chapters of this thesis.

I would like also to thank the past and present members of the AI lab, who have always provided a stimulating and enjoyable place to work. In particular, I must mention the people with whom I have shared offices during my time at Trinity. I have had intense discussions on many subjects (including AI) with Gabriele, Lorcan, Michelle, Marco, Patrick, Rob and Shane. I have learned a lot from these folk and I am proud to consider them my friends. Thanks especially must go to Lorcan Coyle whose insightful views and willingness to help contributed greatly to this project.

It is easy to take your working environment for granted. I am grateful to the support staff and faculty of the Computer Science Department in Trinity College Dublin who have always been helpful to me over the course of my work.

This thesis was completed during the Enterprise Ireland sponsored project on Data Mining and Personalisation in Information Delivery and E Commerce. I am very thankful for the generous financial support of Enterprise Ireland throughout this project. Thanks to Joe Curtis, John Fagan for their careful stewardship and friendly support, and to Carol Gibbons for always being so helpful, pleasant and thoroughly professional in her dealings with us.

I owe a debt of gratitude to my family who have always been completely understanding of the fact that their son/brother never seems to have left school! Thanks Mum, Dad, David, Maria and Suzanne.

Finally, thanks to Michelle for her love and patience.

Abstract

The success of file-sharing networks demonstrates that there is a huge potential market for digital music services, if the music industry can find a service model that is attractive to listeners. The concept of digital distribution is appealing to the music industry, because it suggests the possibility of direct access to the consumer. In this thesis, we present Smart Radio, a prototype for an alternative distribution approach, one in which the consumers themselves are producers, building playlists of music which can be streamed to like-minded listeners. In this approach the service provider is an enabler of community and social processes, allowing information and recommendations to flow between users of the system, rather than engaging in targeted marketing.

However, developing recommendation and personalisation techniques for music systems is problematic. Whereas information retrieval systems use keyword extraction to represent the subject matter of a document for indexing and retrieval, the comparable process for audio artefacts is still a subject of much research. The alternative is to rely on human input for content-description which can be a knowledge-intensive process. Automated Collaborative Filtering (ACF) is a technique which uses user ratings as a means of countering the problem of content elicitation. One serious drawback with ACF is that it is not sensitive to a user's interests at a particular moment. ACF recommendation sets typically contain a mix of items reflecting the user's accrued interests, rather than localised interests. We solve this problem using a Case-Based Reasoning (CBR) technique, in which we rank ACF recommendations according to their similarity to the user *context*. We call this technique *context-boosted ACF*. We define the context using a case-like representation of the user's last playlist. The case representation describes the *composition* of a playlist in terms of the basic meta-tags we have been able to extract from each mp3 file associated with the playlist. We consider the light-weight case representation we use to be appropriate for an extension to ACF, which is often used in domains where content description is scarce.

Increasingly, there has been a demand for objective evaluation criteria for automated recommendation systems. Conventional off-line techniques are not suitable for directly comparing recommendation strategies that produce different types of ranking, such as ACF and context-boosted ACF. We introduce a novel on-line evaluation technique in which two algorithms simultaneously compete to provide the user with recommendations. Unlike the off-line analysis, the on-line methodology judges the relative degree of success of each strategy according to whether recommendations were used by the on-line user. Our experiment, conducted over 15 weeks, found context-boosted ACF to be significantly more popular than ACF for all usage ranges.

We demonstrate that the ACF mechanism is similar to an interactive CBR technique called *case completion* in which an incomplete target query is incrementally filled out in an interactive

dialogue with the user. Using this insight we address some of the problems inherent in ACF systems such as query efficiency and memory overhead by applying techniques used in CBR. We implement our ACF component using a Case Retrieval Net (CRN) which reduces memory overhead by storing each feature-value pair once. We demonstrate how the calculation of the Pearson coefficient can be distributed among the nodes of the CRN. It has been recognised in ACF literature that the use of this measure is a computational bottleneck when calculating neighbours from a large set of users. Accordingly, we showed how retrieval time is speeded up because local similarity calculations are performed once for each node and distributed to indexed cases. Given that the structure of CRNs can be adapted for parallel processing, we suggest that this technique can be extended for large ACF data sets.

Table of Contents

LIST OF FIGURES	XI
GLOSSARY	XV
ASSOCIATED PUBLICATIONS	XVI
CHAPTER 1 : INTRODUCTION.....	1
1.1 THE PROTOTYPE FOR THE DIGITAL ECONOMY	2
1.2 CONTENT REPRESENTATION	2
1.3 CASE-BASED REASONING	4
1.4 AUTOMATED COLLABORATIVE FILTERING	6
1.4.1 ACF and timeliness	8
1.5 SMART RADIO	8
1.6 CONTRIBUTIONS OF THIS THESIS	9
1.6.1 ACF and CBR.....	9
1.6.2 Context-Boosting ACF	10
1.6.3 On-line Evaluation	11
1.6.4 Community-Based Recommendation System	11
1.7 SUMMARY AND STRUCTURE OF THE THESIS.....	12
CHAPTER 2 : THE DISTRIBUTION OF MUSIC ON-LINE	15
2.1 MP3 KILLED THE RADIO STAR.....	16
2.1.1 What is a File-sharing Network/P2P Network?.....	18
2.1.2 The MP3.....	19
2.1.3 Techniques of Music Delivery.....	20
2.2 PHYSICAL DISTRIBUTION	21
2.3 ON-LINE DISTRIBUTION.....	23
2.3.1 On-line Distribution Model.....	24
2.3.2 Content Stake Holder	25
2.3.3 Services on Offer	28
2.3.4 Discussion	31
2.4 VALUE-ADDED SERVICES	32
2.4.1 Personalisation	32

2.5 SEARCH AND PERSONALISATION	33
2.5.1 Personalisation	34
2.5.2 Content-Based Music Retrieval.....	35
2.5.3 Automated Signal Analysis.....	35
2.5.4 Human Analysis	37
2.5.5 Categories of Human Mark-up	40
2.5.6 'Content-less' Retrieval	42
2.5.7 Knowledge Expense	42
2.5.8 Review of Music Personalisation Systems	43
2.6 INTRODUCTION TO SMART RADIO	45
2.6.1 Streaming Service	45
2.6.2 Features	46
2.6.3 Editing a Playlist.....	48
2.6.4 Rating Items or Artists	49
2.6.5 Smart Radio Music Library.....	50
2.6.6 Copyright	50
2.7 CONCLUSION	51
CHAPTER 3 : CASE-BASED REASONING	53
3.1 ROOTS OF CBR: AI AND COGNITIVE SCIENCE	53
3.2 CBR KNOWLEDGE	54
3.2.1 Knowledge Containers.....	55
3.3 METHODOLOGY.....	57
3.4 CASE REPRESENTATION, SIMILARITY AND RETRIEVAL	58
3.4.1 Case Representation.....	58
3.4.2 Similarity and Retrieval	58
3.5 REUSE	64
3.6 REVISE.....	65
3.7 RETAIN	65
3.7.1 Off-Line Learning.....	66
3.8 INTERACTIVE CBR	67
3.8.1 Incremental CBR (I-CBR).....	67
3.8.2 Conversational CBR.....	67
3.8.3 Case Completion	68
3.8.4 Case Retrieval Nets.....	69
3.9 CASE-BASED USER PROFILING	71
3.9.1 PTV.....	71
3.9.2 CASPER	73

3.9.3 <i>Dietorecs Travel Assistant</i>	74
3.9.4 <i>Perspective on CBR in the Music Domain</i>	75
3.10 CONCLUSION	75
CHAPTER 4 : AUTOMATED COLLABORATIVE FILTERING.....	77
4.1.1 <i>Early Collaborative Filtering</i>	78
4.2 FILTERING VS. RECOMMENDING	79
4.3 FILTERING BASICS	80
4.4 PRINCIPLES OF ACF	81
4.4.1 <i>Introductory Ideas</i>	81
4.4.2 <i>Data Representation</i>	82
4.4.3 <i>ACF as a Classification Task</i>	84
4.5 TYPES OF ACF.....	84
4.5.1 <i>Model-Based Algorithms</i>	85
4.5.2 <i>Memory-Based Algorithms</i>	87
4.6 ALGORITHMIC CHOICE ISSUES.....	88
4.6.1 <i>Lazy and Partially Lazy</i>	89
4.6.2 <i>Similarity Indexing</i>	89
4.6.3 <i>Dimensionality Reduction</i>	90
4.6.4 <i>Lazy Similarity Measures</i>	91
4.6.5 <i>Similarity Issues</i>	93
4.7 EXTRACTION OF CANDIDATE ITEMS.....	95
4.8 PRODUCING PREDICTIONS	95
4.8.1 <i>Top N Predictions</i>	95
4.8.2 <i>Most Frequent Item Recommendation</i>	96
4.9 ACF ISSUES.....	96
4.9.1 <i>Recommendation Latency</i>	96
4.9.2 <i>Bootstrapping the System</i>	96
4.9.3 <i>The New User</i>	97
4.9.4 <i>The Grey Sheep</i>	97
4.10 COMBINING COLLABORATIVE AND CONTENT-BASED TECHNIQUES.....	97
4.10.1 <i>Combination Methods</i>	98
4.11 DATA ORDERING AND CONTEXT INSENSITIVITY	100
4.12 OBSERVATIONS ON ACF AND CBR	100
4.13 ACF SYSTEMS	100
4.13.1 <i>GroupLens</i>	100
4.13.2 <i>Ringo</i>	101
4.13.3 <i>Bellcore Video Recommender</i>	101

4.13.4 <i>Fab</i>	102
4.14 CONCLUSION	102
CHAPTER 5 : RECOMMENDATION IN SMART RADIO	104
5.1 THE PLAYLIST	105
5.2 RECOMMENDING PLAYLISTS	106
5.2.1 <i>Simple ACF Strategy</i>	106
5.3 CONTEXT BOOSTING ACF	110
5.3.1 <i>Context</i>	111
5.3.2 <i>Context in Smart Radio</i>	112
5.3.3 <i>Case Representation</i>	113
5.3.4 <i>Feature Weights</i>	115
5.4 INTEGRATING CONTEXT RANKING AND ACF	121
5.4.1 <i>ACF/Content-Based Cascade Architecture</i>	121
5.4.2 <i>MAC/FAC</i>	122
5.4.3 <i>Presenting Recommendations</i>	123
5.4.4 <i>A Sliding Window to Cater for Negatively Rated Playlists</i>	124
5.4.5 <i>Recommendation Timing Issues</i>	124
5.4.6 <i>Refractory Period</i>	125
5.5 DEALING WITH ACF LATENCY	125
5.5.1 <i>Selecting Trusted Neighbours</i>	125
5.5.2 <i>Querying by Playlist</i>	126
5.5.3 <i>Explicit Search</i>	127
5.5.4 <i>Fall Back Strategies</i>	130
5.6 CONCLUSION	130
CHAPTER 6 : SYSTEM AND DATA ARCHITECTURE	132
6.1 SYSTEM ARCHITECTURE	132
6.2 WEB SERVICE	133
6.3 SCHEDULING COMPONENT	135
6.3.1 <i>Top Playlists Update</i>	135
6.3.2 <i>Data Recompilation</i>	135
6.3.3 <i>Recommendation Server Update</i>	136
6.3.4 <i>Recommendation Update</i>	136
6.4 DATA COLLECTION AND THE DATA MANAGER	136
6.4.1 <i>Implicit Track Data</i>	137
6.4.2 <i>Implicit Artist Data – Dimensionality Reduction</i>	137
6.5 A SHORT HISTORY OF BOOTSTRAPPING IN SMART RADIO	138

6.5.1 Community Workers.....	141
6.5.2 New Playlists.....	141
6.5.3 Bootstrapping New Users	143
6.5.4 Bootstrap Experiment: Given-n	144
6.6 RECOMMENDATION MODULES	145
6.6.1 A CBR View on ACF	146
6.6.2 Using ACF in a Case Retrieval Net	148
6.6.3 A Context Analysis Module	153
6.6.4 Case-Based Reasoning Module.....	154
6.6.5 Recommender Services.....	155
6.7 CONCLUSION	160
CHAPTER 7 : EVALUATING SMART RADIO	162
7.1 OFF-LINE EVALUATION	163
7.1.1 Evaluation Set-up: Leave-One-Out.....	163
7.1.2 Baseline	164
7.1.3 Comparison with MovieLens.....	164
7.1.4 Smart Radio Dataset Configurations	166
7.1.5 Discussion on Dataset Configurations.....	166
7.1.6 Evaluation	169
7.1.7 Baseline Evaluation	170
7.1.8 ACF Evaluation.....	170
7.1.9 Prediction Correlation.....	177
7.1.10 Dimensionality Reduction	178
7.1.11 Conclusions from Off-line Analysis	179
7.2 ON-LINE EVALUATION.....	180
7.3 RECOMMENDATIONS VS. OTHER SOURCES OF PLAYLISTS.....	180
7.3.1 Weekly analysis	182
7.3.2 User Trends.....	183
7.4 ACF VS. CONTEXT-BOOSTED ACF	185
7.4.1 On-line Evaluation Methodology.....	186
7.4.2 Evaluation Environment.....	187
7.4.3 Results	188
7.4.4 Conclusions from On-line Analysis.....	191
7.5 CONCLUSION	192

CHAPTER 8 : CONCLUSIONS AND FUTURE WORK	193
8.1 OVERVIEW	193
8.2 CONTENT ISSUES.....	193
8.3 EXTENDING ACF.....	194
8.4 ON-LINE EVALUATION.....	195
8.5 A CASE-BASED VIEW OF ACF.....	195
8.6 PERSPECTIVE AND FUTURE WORK	196
8.6.1 <i>Music Database</i>	196
8.6.2 <i>Playlist Maintenance</i>	197
8.6.3 <i>Collaborative Maintenance</i>	198
8.6.4 <i>ACF User Maintenance</i>	198
8.6.5 <i>Diversity</i>	199
8.6.6 <i>Adaptation</i>	199
8.7 SUMMARY	199
REFERENCES.....	201
APPENDIX A: SERVICES PROVIDED BY DIGITAL SERVICE PROVIDERS.....	220
APPENDIX B: SERVICES PROVIDED BY MUSIC RECOMMENDER COMPANIES....	221

List of Figures

Figure 1.1: Transformation mappings in CBR.....	5
Figure 1.2: A schema of the thesis structure	14
Figure 1.1: The client-server architecture	19
Figure 1.2: The figure illustrates the decentralised architecture of a P2P network.....	19
Figure 1.3: Client-side buffer for a streaming application	21
Figure 1.4: The music value chain for off-line distribution (Durlacher 2001).....	21
Figure 1.5: The breakdown of the price of a CD	23
Figure 1.6: The figure illustrates the value chain for digital music distribution (OC&C 2001)	25
Figure 1.7: Three relationship models	29
Figure 1.8: The feature extraction and indexing process used for Internet search engines.....	34
Figure 1.9: Personalisation requires an analysis of the user's behaviour in order to anticipate future possible requests	35
Figure 1.10: AMG allows visitors to contribute content description to their site	41
Figure 1.11: An excerpt from one of the screens offered to users by the Moodlogic client application.....	41
Figure 1.12: Three techniques typically used to procure description for retrieval of audio artefacts	42
Figure 1.13: User preferences represented extensionally in the AgentArts software	45
Figure 1.14: A screenshot of a Smart Radio home page.....	46
Figure 1.15: Smart Radio playlist options.....	47
Figure 1.16: Smart Radio controls	48
Figure 1.17: A playlist in <i>Edit</i> mode.....	48
Figure 1.18: Choosing the <i>artist-replace</i> icon to remove one David Gray song and replace it with another.....	49
Figure 1.19: The artist rating pop-up box	50
Figure 1.1: The transformation mappings used in CBR.....	55
Figure 1.2: The knowledge containers used in the CBR process.....	56
Figure 1.3: The CBR-Cycle from Aamodt and Plaza (1994).....	57
Figure 1.4: Defining similarity measures for the domain of PC sales (from Stahl 2002).....	60
Figure 1.5: An example of a <i>k</i> -D-tree for CBR (from Wess et al. 1993).....	62
Figure 1.6: Query relaxation	63
Figure 1.7: The adaptation continuum (based on Wilke, Smyth & Cunningham 1998).....	64
Figure 1.8: The interactive CBR process.	69

Figure 1.9: The CRN structure (from Lenz & Burkhard 1996)	70
Figure 1.10: The PTV rating system	72
Figure 1.11: The concept trees used for similarity calculation in CASPER (from Bradley et al. 2000)	73
Figure 1.12: On the client side, CASPER uses <i>k</i> -NN to classify candidate jobs for the user (Bradley et al. 2000)	74
Figure 1.13: An example of a case in the Dietorecs travel assistance system	74
Figure 1.1: A content-based filter	80
Figure 1.2: ACF takes advantage of the overlap in interests within communities of users	82
Figure 1.3: A decision tree for whether a user has listened to the artist Leonard Cohen	86
Figure 1.4: ACF is a cyclical, interactive process	88
Figure 1.5: The Venn diagram illustrates the problem of measuring similarity based on few overlapping ratings	93
Figure 1.1: A playlist built by Smart Radio user, <i>coylel</i> , dedicated to another user, <i>Patrick</i>	106
Figure 1.2: The novelty preference in Smart Radio	108
Figure 1.3: The ACF algorithm assumes that items of all types (coloured bars) are equally important at time t^n	110
Figure 1.4: ACF primes a portion of the Case Base. Primed cases are ranked by similarity to the user's context.	112
Figure 1.5: A playlist represented in terms of its genre/artist composition	113
Figure 1.6: A target playlist and three playlists with differing degrees of similarity	115
Figure 1.7: The figure illustrates the similarity between the target playlist and three other playlists	119
Figure 1.8: The target playlist and three playlists represented in terms of their <i>genre_</i> <i>compositions</i>	120
Figure 1.9: A schema of the playlists indexed using a case retrieval net	122
Figure 1.10: The darker shaded cases in the third stage indicate cases which best match the listener's current listening context	123
Figure 1.11: The top five playlists at any time are context-sensitive playlists	124
Figure 1.12: The new playlists panel in Smart Radio	126
Figure 1.13: The result set from a 'query by playlist'. The top playlist is displayed by default in the right panel.	127
Figure 1.14: The playlist query panel in Smart Radio	127
Figure 1.15: A result set for the playlist query: 'artist: dylan, springsteen, u2'	128
Figure 1.16: Playlist marked for manual adaptation	129
Figure 1.17: Using the edit function the listener can replace flagged tracks with another by the same artist or within the same genre	129
Figure 1.1: The primary components in the Smart Radio system	132

Figure 1.2: The system architecture from the perspective of the web service	134
Figure 1.3: A demonstration of the problem of using ACF when users have not yet rated many items	139
Figure 1.4: Users can select those neighbours whose new playlists they wish to view	140
Figure 1.5: Smart Radio users can view new playlists compiled by selected neighbours	141
Figure 1.6: An illustration of whether users compiled new playlists or played previously compiled lists	142
Figure 1.7: Users in Smart Radio rely upon the industry of a few prolific playlist compilers.....	143
Figure 1.8: On registering, users are encouraged to rate a number of artists so that they can immediately start receiving recommendations.....	144
Figure 1.9: During the bootstrap process artists can be chosen from 21 different genres	144
Figure 1.10: The performance of track data vs. artist data during the bootstrap phase using the <i>Given-n</i> technique.	145
Figure 1.11: A comparison of the CBR case completion process with the ACF recommendation cycle	147
Figure 1.12: The architecture of a CRN using the user profiles, M, N and P and a set of ratings (1–5) for items <i>A, B, C, D</i>	149
Figure 1.13: The first stage of activation. The nodes in black illustrate the activation of the target IEs.	151
Figure 1.14: The second stage of activation. Activation is propagated through interconnected IEs.	151
Figure 1.15: The final stage of CRN activation. The relevance function collects the activation in each connected, activated IE.	152
Figure 1.16: The mean time per prediction for a flat neighbour search vs. a CRN-based neighbour search	153
Figure 1.17: An example of the sliding window used to collect context data	154
Figure 1.18: A schema of the playlists indexed using a CRN.....	154
Figure 1.19: A flow chart demonstrating two asynchronous process flows	157
Figure 1.20: A synchronous request updates the recommendation database in response to a playlist being played	159
Figure 1.21: Each recommended playlist is accompanied by an explanation.....	159
Figure 1.22: The <i>Query by Playlist</i> facility allows users to request playlists <i>similar</i> to the current playlist.....	160
Figure 1.1: A plot of neighbourhood size vs. Mean Absolute Error	165
Figure 1.2: The graph plots mean neighbourhood size as the maximum neighbourhood size, <i>k</i> , is increased.....	171
Figure 1.3: MAE vs. maximum neighbourhood size, <i>k</i>	172

Figure 1.4: MAE for 70 users in the <code>track_explicit</code> dataset for $k = 50$. We could not make predictions for one user.....	173
Figure 1.5: MAE for 70 users in the <code>artist_implicit_explicit</code> dataset for $k = 50$	173
Figure 1.6: This graph overlays the graphs in Figure 1.4 and Figure 1.5	174
Figure 1.7: k vs. percentage of items predicted.....	177
Figure 1.8: k vs. the mean time per prediction.....	179
Figure 1.9: The source of playlists played in Smart Radio during the evaluation period	182
Figure 1.10: Source of playlists: weekly trends	183
Figure 1.11: The graph illustrates that the majority of users were <i>light</i> users, playing from 1 to 10 playlists	184
Figure 1.12: The breakdown per usage range of the 1012 playlists played during the evaluation period.	184
Figure 1.13: The proportions of playlist sources in each usage range	185
Figure 1.14: A screenshot illustrating an interleaved recommendation set	188
Figure 1.15: The cumulative scores for the ACF vs. context-boosted ACF analysis	188
Figure 1.16: ACF vs. context-boosted ACF over a 15-week period.....	189
Figure 1.17: ACF vs. context-boosted ACF divided into ranges of usage.....	190
Figure 1.18: The proportion of ACF to context-boosted ACF per user during the evaluation period	191

Glossary

ACF	Automated Collaborative Filtering
AI	Artificial Intelligence
CBR	Case-Based Reasoning
CRN	Case Retrieval Net
<i>k</i>-NN	<i>k</i> -Nearest Neighbour

Associated Publications

Journals

Context Boosting Collaborative Recommendation (2004), Hayes, C., Cunningham, P., to appear in the Journal of Knowledge Based Systems, special issue AI2003, Volume 17, Issue 5-6, July 2004, Elsevier.

Community Based Music Radio (2000), Hayes, C., Cunningham, P., Journal of Knowledge Based Systems, special issue ES2000, Volume 14, Issue3-4, June 2001, Elsevier.

Peer Reviewed Conference/Workshop Proceedings

Context Boosting Collaborative Recommendations (2003), Hayes, C., Cunningham, P., to appear in Applications and Innovations in Intelligent Systems XI, proceedings of the Twenty-Third Annual International Conference of the British Computer Society's Specialist Group on Artificial Intelligence (SGAI), Cambridge. Springer Verlag.

Programme-Driven Music Radio (2002), Hayes, C., Cunningham, P., Clerkin, P., Grimaldi, M., in the proceedings of the European Conference on Artificial Intelligence ECAI, 2002, editor: Frank van Harmelen, IOS Press.

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

Smart Radio - Building Music Radio on the Fly (2000), Hayes, C., Cunningham, P. in Applications and Innovations in Intelligent Systems VIII, proceedings of ES2000, The Twentieth SGES International Conference on Knowledge Based Systems and Applied Artificial Intelligence. eds. Macintosh A., Moulton, M., Coenen, F. (Springer-Verlag).

An On-line Evaluation Framework for Recommender Systems (2002), Hayes, C., Massa, P., Avesani, P., Cunningham, P., in the proceedings of the Workshop on Recommendation and Personalization in eCommerce at AH2002, 2nd International Conference on Adaptive Hypermedia and Adaptive Web Based Systems.

Distributed CBR using XML (1998), Hayes, C., Cunningham P. and Doyle, M. in proceedings of the Workshop for Intelligent Systems and Electronic Commerce as part of the German Conference on Artificial Intelligence, Bremen 1998 (KI-98).

Shaping a CBR View with XML (1999), Hayes, C. and Cunningham, P., in Case-Based Reasoning Research and Development, proceedings of the Third International Conference on Case Based Reasoning, Seeon Germany (Springer- Verlag), 1999

Representing Cases for CBR in XML (2002), Coyle, L., Hayes, C., Cunningham, P., In proceedings of the UK CBR Workshop 2002. Part of twenty-second Annual International Conference of the British Computer Society's Specialist Group on Artificial Intelligence (SGAI), Cambridge.

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

Automated Case Generation using Knowledge Discovery (2001), Clerkin, P., Hayes, C., Cunningham, P., techniques in proceedings of the workshop on CBR in e-commerce, Fourth International Conference on Case Based Reasoning, Vancouver BC, Canada, July 2001.

A Case-Based Personal Travel Assistant for Elaborating User Requirements and Assessing Offers (2002), Coyle L, Cunningham P., Hayes C., in proceedings of the 6th European Conference on Case-Based Reasoning, (2002), Aberdeen, Scotland.

Smart Radio - Building Smart Music Radio (1999), Hayes, C. in Proceedings of AICS'99, Irish Conference of Cognitive Science and Artificial Intelligence.

Concept Discovering in Collaborative Recommender Systems (2003). Clerkin, P., Cunningham, P., Hayes, C., in proceedings of the 14th Irish Conference of Artificial Intelligence and Cognitive Science (AICS 2003), Trinity College Dublin, September 2003.

Technical Reports

CBR Net: Smart Technology over a Network (1998), Hayes, C., Doyle, M., Ferrario M., Cunningham, P. and Smyth, B.. TCD technical report. Featured paper in ai-cbr hall of fame. <http://www.aicbr.org/hall.html>. Available at <http://www.cs.tcd.ie/publications/tech-reports/reports.98/TCD-CS-1998-07.ps>

Smart Radio - an Application Proposal (1999), Hayes, C. Technical report, Department of Computer Science, Trinity College Dublin Available at <http://www.cs.tcd.ie/publications/tech-reports/reports.99/TCD-CS-1999-24.pdf>

Chapter 1: Introduction

In August 1953 a young Tennessee truck driver stepped up to the microphone in the small recording booth at Sun Studios, 706 Union Avenue, Memphis. On a whim, Elvis Presley had decided to record two songs by the black vocal harmony group, The Ink Spots as a present for his mother's birthday. Earlier that year, the studio owner, Sam Philips, a little known radio announcer who had set up a modest sideline recording local blues artists, had turned to his secretary and declared, "if I can only find a white man with the negro sound and the negro feel, I could make a billion dollars".

In 1953, the popular music scene in the USA was made up of three main styles of music – Pop, Country & Western, and Rhythm & Blues, each with its own separate performers, record labels and audiences. Record sales in the US stood at approximately \$213 million. By August, 1955 after 5 successful chart singles, the hybrid of pop, country and blues recorded at Sun studios by Elvis Presley had ushered in what has become known as the *Rock and Roll revolution*, a sweeping away of older popular music forms, music markets and cultural mores. By 1959 record sales in the US stood at \$603 million, with *Rock and Roll* constituting 42.7 % of the market. Sam Philips, a businessman at heart, had sold his interest in the Elvis Presley recordings in 1955 to RCA, one of the four 'majors' for a record \$40,000. However, by 1959 the major record labels in the US suffered a 34% drop in market share at the hands of regional independents that had pioneered the new music.¹

Fast forward to August 2003, exactly 50 years after Elvis Presley first walked in to a studio. Another revolution is taking place which is affecting the mainstream music industry. By typing Presley's name into the search field of an Internet file-sharing client, it would appear that every song ever recorded by the 'King of Rock and Roll' is freely available for download from a network of globally distributed computers. In the same month, the Recording Association of America (RIAA) reports that the half yearly sales returns for 2003 have continued to follow the downward trend observed since 1999². The RIAA, which represents companies like BMG, the current owners of Elvis's Sun recordings, has been locked in litigation with file-sharing networks and individuals to try to prevent the unauthorised distribution of copyrighted recording, which it claims is the source of the slow down. Commentators suggest that *the digital revolution*, which has facilitated unprecedented access to information, and now music, has yet to fully work itself out and will ultimately change forever the way music is made, distributed and consumed.

¹ Scaruffi, Piero (2003) The History of Rock Music 1955-1966. Available at <http://www.scaruffi.com/history/cpt11.html>

² RIAA (2003). RIAA Releases 2003 Mid-Year Shipments. <http://www.riaa.com/news/newsletter/082903.asp>

1.1 The Prototype for the Digital Economy

The distribution of music on-line has been cited as a prototype model for the digital economy (Birch & Davidson 2001). Many music industry insiders had considered it to be ‘the next big thing’, in the industry parlance, which would allow record companies to increase profits by marketing directly to their clients (Birch & Davidson 2001). Like Sam Philips, 50 years ago, the labels were well aware of a golden opportunity; in this case it relied upon figuring out how to fuse their centralised distribution model with a secure means of digital delivery.

Unfortunately they delayed too long. The industry was walking on untested ground and was hesitant about compromising their core asset – ownership of recording content. Coming up with distribution and payment models, royalty systems that would satisfy artists and publishers, and a secure means of delivery was never going to be an easy task. However, the advent of the Napster file-sharing network in 1999 heralded a plethora of similar P2P networks, non-centralised networks which enabled people to directly share digitally compressed music files. Since 1999, millions of users of file-sharing networks have swapped billions of music files, which in terms of the numbers of songs, is estimated at equivalent to the number of CDs sold in the same period (Liebowitz 2003). The ‘next big thing’ for the millions of users of P2P systems is the availability of the music itself, 50 years or more of recordings at their fingertips. The music industry has responded with litigation and more recently with the introduction of its own music subscription services.

In this thesis we are concerned with how music should be distributed online. We describe Smart Radio, a service for delivering *playlists* of music, which has been in operation in the Computer Science Department at Trinity College for three years. Several reports identify value-added features as the key for content providers to coax users from file-sharing networks. We suggest that Smart Radio’s community-based personalisation offers an alternative distribution paradigm for content providers, one in which listeners are empowered to affect the flow of recommendation information across the networked community. However, personalisation in the music domain is difficult because good content description is expensive to obtain. The most common way of dealing with this is to use Automated Collaborative Filtering (ACF) in which no explicit description of the music is required. A significant drawback to ACF is that it is insensitive to the listener’s current preferences. We describe how we solve this problem using a knowledge-light Case-Based Reasoning implementation. Furthermore we show how ACF can be usefully implemented as a type of CBR system. Finally we present a novel on-line evaluation of our techniques.

1.2 Content Representation

Typically personalisation systems operate by automatically building a user model which is a representation of the user’s interests. The user model is based on a description of things the user has liked (or disliked) in the past. The goal is to anticipate the items the user will like in the future

based on information stored in the user model. However, one of the biggest obstacles to music personalisation is the difficulty of obtaining content description on which to base a user model in the first place. There are two ways of assigning content description to music files.

1. Feature Extraction using signal analysis
2. Human Analysis

Feature extraction using signal analysis can be contrasted with the feature extraction process used in textual information retrieval systems (Foote 1999). While a page of text can be represented by a vector of words, and indexed for retrieval, the ‘semantic’ components of an audio file are not *a priori* available. Signal analysis usually proceeds using a Fourier analysis of the wave forms of the audio signal. However, this produces a very low level representation. While signal analysis conducted using psychoacoustics (models of human auditory perception) produce higher-level representations, these representations are still far from interpretable from a human point of view. The chief drawback, however, to automated signal analysis is that humans bring additional perceptions to bear upon a piece of music which cannot be extracted by wave form analysis.

The second means of marking up music is by human analysis. We identify two broad types: Formal and Informal systems of description. Formal description systems for music are based in various disciplines such as music theory, music-cognition, ethnomusicology and psychology. While these disciplines attempt an objective analysis of music, the descriptions often cannot be represented outside the bounds of that discipline itself.

Informal systems of music description refer to forms of music knowledge which are not based in an academic or formal framework, although they still may be part of an organising semantic infrastructure. Informal knowledge is derived from the place of the musical artefact in human society and culture. As such it is subject to the value systems currently being operated. Informal music description is characterised by a more subjective appraisal of the music item. An example of an informal system of music description is the mark-up provided by a music site like Allmusic.com, where the descriptors offered are impressionistic describing moods, emotions and sensations. For example, one category of music is described as “dark, pessimistic, bitter” (See Figure 2.11). The same site also provides information on which artists and bands are similar, or are considered to have influenced other bands. This information is provided by Allmusic.com’s team of over 900 music contributors³. The knowledge elicitation techniques we have described up to this point can be considered to be expensive processes. (i.e. that require a great deal of human authoring, editing or guidance).

However, the least expensive means of collecting music ‘mark-up’ in a distributed environment is by collecting user ratings (Shardanand & Maes 1995). This is an informal technique in which users are simply required to rate a music item according to a numeric scale and these scores are then collected by the service provider (see Table 1.1). Ratings can be submitted

^{3 3} <http://www.allmediaguide.com/data.html>

explicitly or can be derived implicitly from the user’s on-line behaviour. This type of usage content is generally referred to as ‘content-less’, in contrast to mark-up where clearer semantics are attached. While this technique still requires human input, the data can be collected in a distributed fashion as part of a deployed application.

Table 1.1: The table illustrates the typical format of usage data

	User A	User B	User C	User D	User E
Love me tender	0.8	0.2	-	1.0	0.8
Blue suede shoes	0.2	0.8	0.6	0.4	-
Suspicious minds	0.6	0.4	0.2	-	0.8

A distinction is often made in the literature between ‘content-based’ and ‘content-less’ approaches to recommendation (Balabanovic & Shoham 1997). The distinction is made because there are no formal semantics associated with the ratings given to a music item. We will suggest that this distinction is not entirely accurate, and that rating data, although noisy, implicitly contains meaning with respect to how the music item is valued among a population of users.

Case-Based Reasoning is a successful content-based approach to retrieval. We will briefly describe it in the next section after which we will introduce Automated Collaborative Filtering, a so-called ‘content-less’ approach to recommendation.

1.3 Case-Based Reasoning

Case-Based Reasoning (CBR) is a methodology in artificial intelligence for solving problems by reusing the solutions from previously solved problems. The origins of CBR were stimulated by research into how people remember information and how they are in turn reminded of information (Schank 1982). It was observed that people commonly solve problems by remembering how they solved similar problems in the past.

In CBR systems previous problem-solving episodes are stored as cases in a case base and typically each case has a case specification part and a solution part. In a diagnosis domain, for instance, the specification might describe fault symptoms and fault context and the solution describes the cause of the fault. CBR can also be used in situations where this problem-solving vocabulary is not appropriate. More generally, it can be viewed as a means of determining outcomes associated with situation descriptions. With CBR, instead of attempting to model the causal interactions that link outputs to inputs, the idea is to retrieve and adapt cases when solving new problems. This is described in Figure 1.1 where SP represents a specification of a problem that needs to be solved, SL is a solution to that problem and FP is some hypothetical First Principles reasoning that would infer the appropriate solution for the problem description SP. The idea in CBR is to avoid having to model this First Principles reasoning by instead retrieving a case with a similar description SP' and adapting the solution to that case (SL') to fit the problem in hand. The implication is that this retrieval and adaptation process is simpler to implement than the First Principles reasoning.

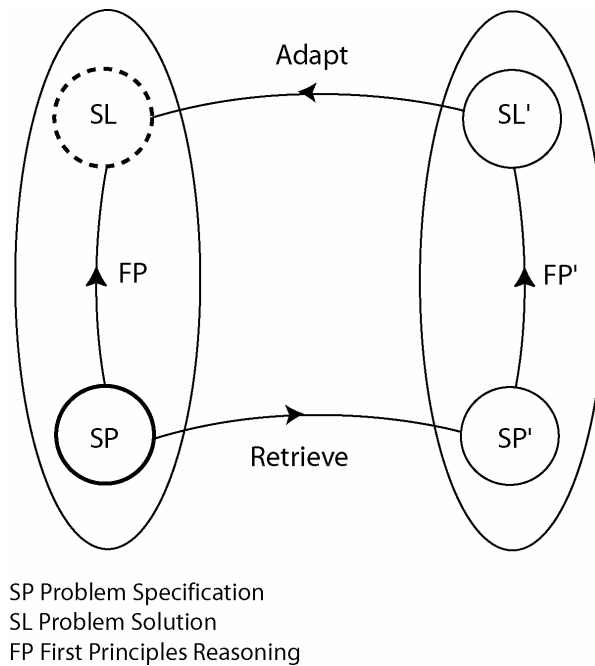


Figure 1.1: Transformation mappings in CBR

Thus CBR is often credited with providing an alternative solution to the intensive knowledge engineering requirements of rule-based or model-based systems. However, for this to hold true the knowledge engineering involved in building a case base must be low. In many situations case data can easily be obtained. This might be in the form of problem-solving episodes recorded in diagnostic logs, or in the form of catalogue or configuration data. In such situations a CBR system can be more quickly deployed than a rule-based system.

However, where the development of a CBR system also involves a deep analysis of the domain in order to produce a case base, the advantage of CBR over knowledge intensive strategies is not so obvious. Despite this, it could be argued that in many situations only a CBR approach is viable. This occurs in ‘weak theory’ domains where it is difficult or impossible to elicit first principle rules from which solutions may be created. Typically, in such domains human experts would reason from precedents rather than from first principle. CBR is a lazy learning technique, deferring reasoning until query time. Thus it is suited to domains where the case base is being added to or edited frequently and where it would be difficult to create a model that accurately reflected the state of the changing environment. The Internet is an example of such an environment, and CBR has found increasing application there as an assistant in Internet commerce stores, as a reasoning agent for technical support and as a strategy for user personalisation. An example of the latter is case-based user profiling, where a case-based user profile captures the interests of each user and makes recommendations based on the similarity of the profile to items such as TV programmes (Smyth & Cotter 1999 a,b), job descriptions (Bradley et al. 2000) or flight offers (Coyle et al. 2000). An example of a case-like profile is given in Table 1.2. This is an example of CBR where the typical problem-solving vocabulary is not appropriate. In this situation the case is

used to locate songs that the user will like based on his/her profile. An interesting feature of this approach is that the target case (the user profile) does not have to be explicitly specified by the user but can be constructed by passively monitoring the user on-line preferences. Although very simple, the representation of the user profile in Table 1.2 and the song in Table 1.3 again raises questions about knowledge elicitation in the music domain since only one feature, tempo, could possibly be extracted automatically. Secondly, since most of the features are symbolic, a domain ontology would be required to encode the local similarity measures.

Table 1.2: A case-like description of a user's interests

User Id	A
Artists:	Carl Perkins, Jerry Lee Lewis
Type:	Male solo, male band
Genre:	Rock'n'Roll, Country, Rock
Style:	Ballad, Rocker
Date:	1954 - 1968
Instrument:	Vocal, electric guitar, piano
Key words:	Betrayal, hurt, love
Tempo:	mid-tempo, fast

Table 1.3: A case-like description of a song

Song:	Love Me Tender
Title:	Love me Tender
Artist:	Elvis Presley
Type:	Male solo
Genre:	Rock'n'Roll, Pop
Style:	Ballad
Album:	Elvis Love Songs
Date:	1956
Instrument:	vocal, acoustic guitar
Key words:	love, romance
Tempo:	slow
Film:	Love me Tender 1956

In the example above, one of the relations that a domain expert might capture is that Elvis Presley, Jerry Lee Lewis, and Carl Perkins all started their careers in the 1950s, recording rock'n'roll records at Sun Records in Memphis, and are thus very similar. However, a relation that would be difficult to capture would be the fact that the song represented in Table 1.3 (a slow, romantic love song) is not typical of the material produced in that genre at that time. For *user A* who prefers classic rock'n'roll this may not be a suitable recommendation. Thus, this strategy is limited by the granularity of the case description and the similarity metrics employed, and cannot make subtle distinctions between items which are similar in feature terms but intuitively dissimilar to the eyes or ears of the user.

A symptom of this problem is that a content-based strategy may over-specialise, not finding new items outside the user profile description that might be of interest. This is a serious problem in the music domain where there is great benefit in suggesting new artists that the user might enjoy based on his/her listening profile. Building such knowledge into a CBR system means further enhancing the case description and the similarity metrics, leading to knowledge engineering overheads, longer retrieval times and larger storage requirements. It could be argued that no case description will ever be fine enough to satisfy the intuitive distinctions people make between apparently similar items.

1.4 Automated Collaborative Filtering

As we suggested at the end of the section 1.2, an alternative to the type of content-based representation we illustrate in Table 1.3 is a so-called 'content-less' representation. This tag refers to the fact that (music) items are marked up, not with a set of descriptors, but with a set of scores.

Each score is a subjective rating by a user of the item in question. These ratings can be explicitly submitted or derived implicitly by monitoring the user’s usage of the item. As we suggested earlier, the knowledge engineering task for such a representation is generally low since it is collected from a distributed set of users, and can be collected as part of the functionality of an already running system. For example, users could be provided with a facility to rate items suggested by a content-based recommender. This would give the type of representation shown in Table 1.4.

Table 1.4: An example of ‘content-less’ representation of 3 early rock’n’roll songs

	Love Me Tender	Blue Suede Shoes	Whole Lotta Shakin’ Goin’ On
User A	0.2	0.8	-
User B	0.8	-	0.4
User C	-	0.2	0.2
User D	0.2	1.0	-

While the description of the song ‘Love me Tender’ in Table 1.4 is not as immediately interpretable as the content-based description in Table 1.3, the description is far from ‘content-less’. The rating data implicitly contains meaning with respect to how the song is considered among a population of users. It is this implicit knowledge which is leveraged by a technique like Automated Collaborative Filtering (ACF) to make recommendations. With rating data, ACF makes use of the subtle distinctions people make when selecting or rejecting from any set of items. This type of knowledge (aesthetic criteria, emotionally or culturally based responses etc.), as we suggested in the last section, is very hard to encode into content-based systems. With the ACF algorithm, it is assumed that these distinctions are systematically encoded with the user’s ratings patterns.

ACF makes recommendations by firstly finding a neighbourhood of users similar to the target user. Similarity is calculated based on the items the users have rated in common. Predictions are then made for each item the target user has not yet encountered using a weighted average of the neighbour ratings for that item. In addition to not having to mark-up assets, a key benefit of ACF is that it can make recommendations that would not have been found by a content-based strategy. Given that a content-based profile contains descriptions of the types of items the user has used in the past, it is limited to making recommendations based on these descriptions. ACF, on the other hand, leverages the experience of a neighbourhood of users who collectively have experienced more items, and thus the user can receive a broader range of recommendations that have been endorsed by like-minded users.

Typically, ACF has used lazy, memory-based techniques (Shardanand & Maes 1995, Resnick et al. 1994). However, the calculation of the neighbourhood set has proven to be a major bottleneck (Herlocker et al. 1999). With ACF being used in large sites, memory-based algorithms need to meet the constraints imposed by normal request response times. In some cases, the neighbours that are selected to meet a request are not necessarily the best, but rather the most similar that could be found by sampling profiles in high speed memory (Herlocker et al. 2000). We

suggest an alternative approach using an indexing technique borrowed from Case-Based Reasoning.

1.4.1 ACF and timeliness

One serious drawback with 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 user's current interests may cause frustration and distrust. Extending our music example, if *user A* has an interest in *Jazz* and *Classical music* as well as *Rock'n'Roll*, an ACF recommender may suggest a classical piece when *user A* is listening to Elvis Presley. While the classical recommendation might be a good recommendation, it is not a *timely* recommendation. One of the topics of this thesis is the introduction of *timeliness* into the ACF recommendation technique. 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.

1.5 Smart Radio

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 playlists from scratch, play past favourite playlists, or choose playlists recommended by their neighbours.

Smart Radio provides a personalised music service based on the usage patterns of each of its listeners. We use a novel hybrid of ACF- and CBR-based techniques to achieve this. The design goal in Smart Radio was to provide a personalised service of streaming music using a recommendation system to suggest suitable compilations of music to listeners. These playlists are put together by individual users and are then recommended to other similarly minded members. By using a playlist of music as our unit of recommendation the work involved in putting together a new compilation of music is distributed to other listeners. A key feature of Smart Radio is that listeners can identify the author behind each recommended playlist. This facility encourages community participation by allowing users to know who their most consistent neighbours are. In this way, our recommendation system promotes rather than replaces social processes (Hill et al. 1995).

1.6 Contributions of this Thesis

This thesis contains an overall description of the Smart Radio music personalisation system which has operated for a number of years within the Computer Science Department, Trinity College. During its development and operation several issues were encountered. The most interesting ones, which constitute the main contribution of this thesis, are

- An analysis of ACF from a CBR perspective which leads to the implementation of ACF as a Case Retrieval Net.
- The development of an integrated *context-boosted* ACF technique to solve the lack of context-sensitive ordering in ACF recommendations.
- An analysis of *on-line* evaluation for testing new recommendation strategies.
- An analysis of how a legal online music service might operate in which the users are in control and the service provider takes the role of enabler.

1.6.1 ACF and CBR

ACF, which uses subjective user ratings as a proxy for content-based mark-up, can be used to make recommendations where content is not readily available. Although ACF and CBR are often cited as being dissimilar, but complementary techniques, we suggest that they have many features in common. ACF uses a lazy, memory-based technique to retrieve a set of nearest neighbours, as does CBR. We show that the ACF mechanism is similar to an interactive CBR technique called *case completion* in which an incomplete target case is incrementally filled out in an interactive dialogue with the user. At each step, the system uses the information in the set of neighbouring cases to suggest possible completion steps to the user which he/she may accept or decline.

We can view the ACF technique as a long-lived case completion technique in which the user profile is gradually elaborated by feedback from the online user. In this view, the system uses the set of neighbouring user profiles to suggest case completion steps (recommendations). Unlike CBR, however, the dialogue between the system and the user is on-going and case (user profile) completion is indefinitely postponed.

We implement our ACF component using a memory structure used in case completion. A Case Retrieval Net reduces memory overhead by storing each feature-value pair once as an information entity (IE) and indexing all cases that contain this information entity. We show how retrieval time is speeded up because local similarity calculations are performed once for each IE and distributed to indexed cases. As an example, we show how the calculation of the Pearson coefficient can be distributed among the nodes of the CRN during retrieval time. It has often been recognised that neighbourhood computation is a major bottleneck for ACF systems (Herlocker et al. 1999). Herlocker et al. (2000) suggest sampling user profiles to reduce the cost of this process. We would suggest that our technique offers the benefits of a complete search and increased speed.

Furthermore, since CRNs can be adapted for parallel processing (Lenz 1999), the technique is scalable.

Data ordering

In case completion scenarios we can make a distinction between systems that use *surface similarity* and systems that involve *structural similarity*. Surface similarity concerns feature value similarity, whereas structural similarity is defined using the relations *between* features, such as ordering constraints. While the data from which an ACF profile is constructed does have an ordering relation (it represents a shallow trace of the user's 'consumption' over time), this relation is not represented in the ACF profile. The ACF algorithm uses surface features only to assess similarity, discarding any semantics associated with the original ordering of the data. Thus, ACF case completion entities (ACF recommendations) are presented in an ad-hoc way without concern for the order of the entities preceding them. In the next section we describe how we make use of the most recent ordering relation in the ACF data to produce context-sensitive recommendations.

1.6.2 Context-Boosting ACF

The lack of consideration for the ordering of the underlying data means that ACF recommendations are presented without consideration for the user's current interests or context. The objective to context-boosted 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. The Smart Radio domain suffers from a deficit of good 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. We use a representation of what the user is currently using as a proxy for a more explicit context description (which cannot be achieved because of poor content representation). Using this we hope to implicitly capture the user's current short-term interests. We use a *case-based* representation in which each playlist is described with features that indicate the *genre/artist* mixture within 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. Thus we are able to produce a ranking based on playlist similarity. We integrate the ACF and similarity-based ranking using a novel implementation of a MAC/FAC architecture in which the result set from this ACF retrieval is refined by the similarity-based ranking.

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 evaluation (as is standard in machine learning and information retrieval), ranking problems such as these are not as easily studied in an off-line manner.

1.6.3 On-line Evaluation

Increasingly, there has been a demand for objective evaluation criteria for on-line recommender systems. This stems from a difficulty in evaluating which recommender algorithm is better, and in judging which criteria to use when making this evaluation. Normally, evaluations 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 efficacy of a recommender strategy like content-boosted ACF in which a ranking is produced based on user actions at a particular time. In order to test our new hypothesis we perform a comparative analysis of how it performs against a pure ACF strategy. We suggest that an evaluation should measure whether real people are willing to act based on the advice of the system. Unlike the off-line analysis, the on-line methodology judges the relative degree of success of each strategy according to whether the recommendations of each strategy were employed by the on-line user. While this technique can only gauge relative user satisfaction with one strategy over another, it does allow the developer to test whether a new strategy is perceived as being useful to the user base.

1.6.4 Community-Based Recommendation System

One of the attractions of file-sharing networks is that there is not centralized control, or influence by the media conglomerates which own the copyrights of the music being swapped. We suggest that the Smart Radio system gives a working example of a music service in which the content provider takes a back seat, and allows the online community to take control of the processes of organisation and recommendation. This is contrary to the model which the digital service providers are currently pursuing which seeks to develop a direct marketing relationship with the user.

Increasingly, in modern capitalist economies the ability to shape meaning is concentrated in relatively few hands (Fisher 2000). One of the benefits of the Internet is its facility to decentralize this semiotic power, allowing users to organise, discuss and shape meaning. An emerging example of this is how *blogging*, reporting by individuals from their own website, is increasingly being accepted as an alternative to traditional journalistic channels. Indeed there have been several studies of how the Internet is an enabler of community-based relations (Fernback & Thompson 1995) and primary oral culture (Fernback 2003). In the context of Internet distribution of music, Fisher (2000) suggests that the impulse to create individual meaning is attendant

“in the ease with which consumers of digital music can consume it, recombine pieces of it, blend it with their own material – in short can become producers”

We propose that by allowing users to compile and swap playlists of music Smart Radio puts in place a facility where users are enabled to become producers. We analyse how the community dynamic works in Smart Radio and find that it is driven by several prolific playlist producers, whom we call community leaders.

The unregulated distribution of music undermines the ability of the music creators to make money. We suggest that a system like Smart Radio offers a compromise arrangement. Like file-sharing networks, it taps into resources made available by other users: in this case, music expertise. Unlike the file-sharing networks where content is available in an unregulated form, the Smart Radio system can operate as a value-added service for a legally operated music provider. We suggest that this type of lightly moderated community would prove attractive to users who value the ethos of the file-sharing networks. From the service provider's point of view, such an arrangement would fit neatly into the architecture of a system such as Listen.com's Rhapsody system. Thus the recording industry could take the role of a content provider, licensing content to independent music services/retailer.

1.7 Summary and Structure of the Thesis

In brief: the next chapter presents an analysis of the online distribution of music and the reasons for using personalised techniques in this domain. Chapters 3 and 4 present a review of two personalisation strategies, case-based reasoning and automated collaborative filtering. Chapter 5 describes how context-boosted ACF is used to recommend playlists in Smart Radio. Chapter 6 describes the design and implementation of the Smart Radio system, and the strategies employed to manage data for the recommender modules. Chapter 7 presents an off-line and on-line evaluation of the system. In the concluding chapter we discuss possible enhancements for the system.

In more detail: Chapter 2 introduces the Smart Radio system in the context of the emerging market for the online distribution of music. The mechanisms required to distribute music online are discussed, and the current techniques for making personalised recommendations of music analysed. We see that a significant problem for making music recommendation is the lack of available content mark-up. The chapter concludes with the view that, instead of trying to replicate the centralized off-line model of distribution, the music industry should make use of the ability of the internet to form self-organising communities. Thus the music industry would become an enabler of person-to-person distribution/recommendation. We suggest that the Smart Radio project is a prototype for this type of activity.

Chapter 3 gives the background to case-based reasoning, a methodology successfully used for personalization tasks. In particular we review case-based user-profiling. We introduce Case Retrieval Nets, a memory structure suited to efficiently storing large amounts of case data where missing case attributes are common.

In Chapter 4 we review automated collaborative filtering, a so-called 'content-less' approach to filtering and recommendation. In Chapter 2 we noted the difficulty in obtaining descriptive mark-up for music items. ACF offers an alternative approach by recommending items based on user rating data which can be explicitly submitted or derived implicitly. However, ACF is not a magic bullet; in order to work it requires a lot of data. It cannot recommend items that have not yet been rated by other users, nor can it produce recommendations until a user has built up a

rating set. These problems can be alleviated by using a content-based filter or retrieval system in parallel. However, we observe that the ACF technique is not sensitive to the user's current interests, and will recommend items that are not appropriate within the users' context. In Chapter 5, we review the idea of context and introduce a hybrid technique called *context-boosted ACF* in which a lightweight case-based strategy provides context-sensitive ordering to the playlist recommendations of the Smart Radio ACF module. We also describe three techniques for managing a domain where previously 'consumed' items may be recommended again.

We describe the overall system architecture of Smart Radio in Chapter 6, concentrating upon the issue of data collection and manipulation to produce timely recommendations to our users. We show that the system depends on the industry of a few prolific playlist compilers, whose names become well known by other users. We show how a simple dimensionality reduction transformation allows us to quickly bootstrap new users into the system. We develop the argument raised in Chapter 4 that ACF can be treated as a type of content-based technique. We explain how we use a Case Retrieval Net to implement the ACF neighbourhood selection process, and show why this type of memory structure is suited to a lazy, data-intensive process like ACF.

Finally, in Chapter 7 we review the types of evaluation required for a recommender system. We suggest we need an *on-line* approach as well as an *off-line* approach to evaluation. For our off-line analysis we use a *leave-one-out* technique to evaluate whether neighbourhood groups can be produced from the data, the prediction errors and the number of items predicted. We introduce a novel *on-line* evaluation to test the efficacy of our context-boosted algorithm. We deploy both strategies simultaneously in Smart Radio and evaluate which is preferred by users. We find that context-boosted ACF was significantly more popular than standard ACF during our 15-week evaluation period.

This thesis is written so that, as much is possible, each chapter tells its own story. Towards this end, we will occasionally restate ideas addressed in earlier chapters in order to make the discussion of a current topic more concrete. Figure 1.2 gives a plan of the thesis structure.

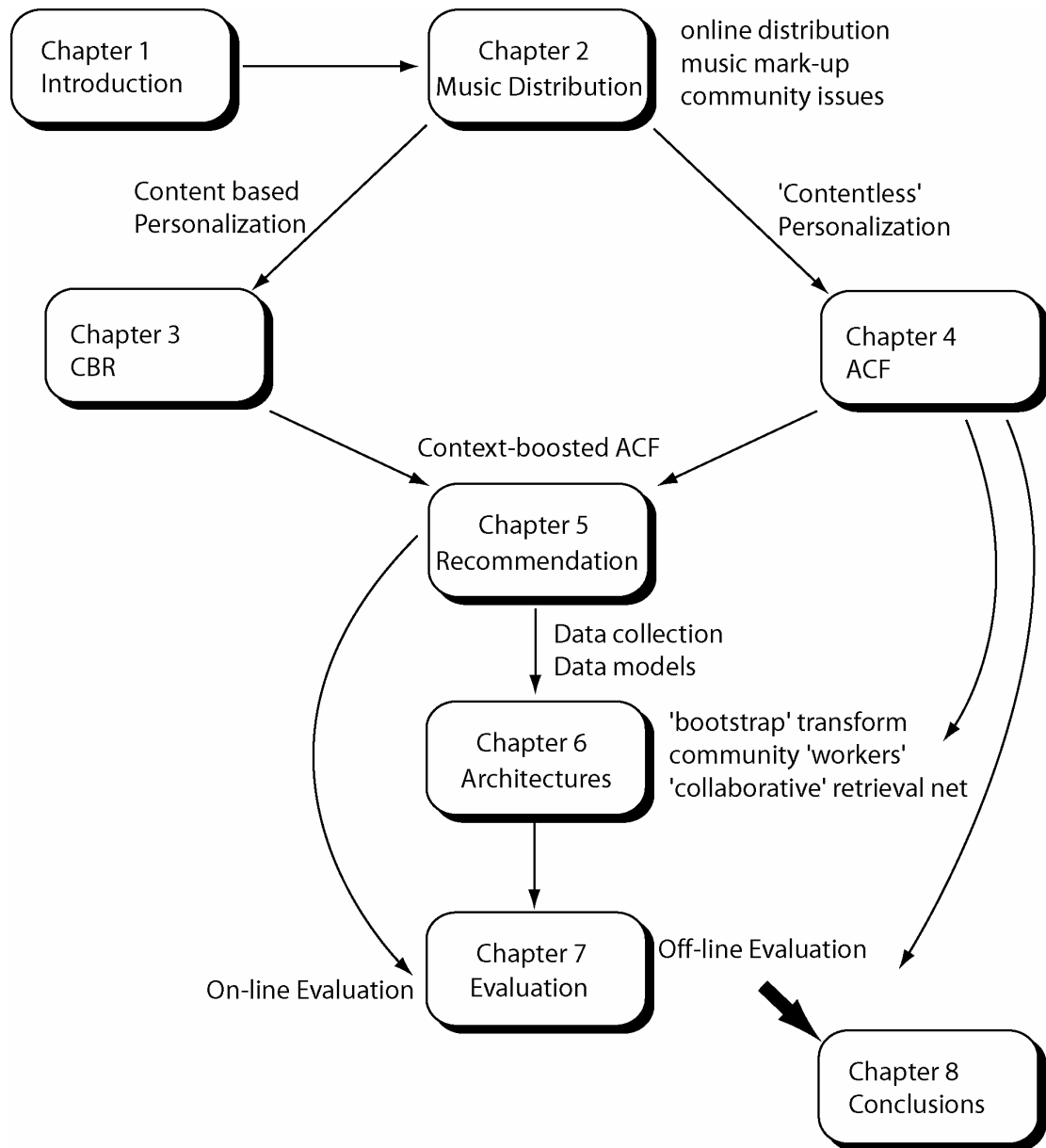


Figure 1.2: A schema of the thesis structure

Chapter 2: The Distribution of Music On-line

All art constantly aspires to the condition of music – Walter Pater, The School of Giorgione

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. Now that 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 warned 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 and content providers. The success of file-sharing networks has forced the hand of the music industry and it has responded on two fronts: litigation, and the promotion of new on-line music services, which are still very much in their infancy.

In this chapter we examine the role of a service like Smart Radio. We compare the value chain for traditional off-line music distribution and new on-line digital distribution models, and suggest that the current initiatives by the music industry are predicated upon maintaining centralised control over the means of production and the means of distribution. We discuss the interests of content providers, creators and consumers in this debate. The music/movie industry has issued warnings about its impending demise whenever new recording media has been introduced such as the videotape and the cassette tape (Frith 1993). However, unlike these situations the Internet has forever changed the manner in which music can be distributed. We look at music industry initiatives to provide legal, secure music services on-line and examine how Smart Radio could be situated as a commercial service. Several reports identify value-added features as the key for content providers to coax users from file-sharing networks. We suggest that Smart Radio adds significant added value in that, like file-sharing networks, it taps into resources made available by other users: in this case, music expertise. Unlike the file-sharing networks where content is available in an unregulated form, the Smart Radio system can operate as a valued added service for a legally operated music provider. The personalisation offered by a service like Smart Radio is transparent, and not open to abuse by the marketing strategies of the content provider. We suggest that this type of lightly moderated community would prove attractive to users who value the ethos of the file-sharing networks.

We examine three of the most successful new music services. We suggest that the experience of operating a personalised streaming service described in this thesis would have many

lessons for these providers. We examine music personalisation technologies being offered by third party vendors. We find that they break down into two categories: knowledge-intensive methods which require the development of complex ontologies of music, and so-called ‘content-less’ techniques which rely upon identifying patterns in the listening data of users. Finally we give an overview of how the Smart Radio system operates.

2.1 MP3 Killed the Radio Star

When we talk about the music industry we generally refer to the interests represented by the five majors (Universal, Warner, Sony, BMG and EMI) who together control about 75% of the global recorded music sales. Although there are many independents, the five majors have enough collective clout to set the agenda regarding the interests of the music industry as a whole. The most vociferous campaigner for the ‘rights’ of the music industry is the Recording Industry Association of America (RIAA) which first came to public prominence during their lawsuit against the Napster file-sharing network.

In February 2001 the US 9th Circuit Court of Appeals issued a ruling against Napster that it had engaged in “knowingly encouraging and assisting” copyright infringement by allowing internet users to search for, and download, copyrighted music files. The court ruled that Napster would have to put in place a filtering system to prevent copyrighted recordings being exchanged. By July of that year, Napster had effectively ceased to operate. During its two-year lifespan millions of people had swapped billions of music files on-line (Birch & Davidson 2002). A survey by the Pew Internet & American Life Project found that at the height of Napster’s popularity users of file-sharing networks did not consider swapping music on-line to be an illegal activity.⁴ In fact, since the demise of Napster, a plethora of new file-sharing networks has sprung up.⁵ A survey by Nielsen/NetRatings placed Kazaa, currently the most popular file-sharing application, as the 6th most popular Internet application in the US, being used by 10.6% of the active on-line population.⁶ The same survey found that Kazaa was one of the ‘stickiest’ applications with the average user spending 2.36 hours per month using it on-line. The music industry, in the guise of the RIAA, has decided that pursuing the new breed of fully decentralised file-sharing networks is fruitless and has responded by suing individuals.⁷

⁴ Downloading Free Music: Internet Music Lovers Don’t Think It’s Stealing. Pew Internet & American Life Project’s On-line Music Report, available at

http://www.pewinternet.org/reports/pdfs/PIP_Online_Music_Report2.pdf, p. 6 (September 28, 2000).

⁵ Napster use slumps 65%. BBC on-line report. <http://news.bbc.co.uk/2/hi/business/1449127.stm>

⁶ More than 72 Percent of the U.S. Online Population Uses Internet Applications. Nielsen/NetRatings (November 2002), available at http://www.nielsen-netratings.com/pr/pr_021218.pdf

⁷ Students sued in piracy battle. BBC News On-line report, 4 April, 2003.

<http://news.bbc.co.uk/2/hi/technology/2917779.stm>

The argument put forward by the music industry is that file-sharing is an infringement of copyright which effectively prevents record labels from making money upon their core asset – content. They illustrate their point by demonstrating trends that show a fall-off in CD sales which they maintain is due to the growth of on-line piracy.⁸ Appealing to the better nature of file-sharing advocates, they claim that file-sharing ultimately hurts artists.⁹ A recent economic analysis concludes that on-line piracy is having an effect on sales but that the RIAA's claims for industry meltdown at the hands of the file-sharing networks is unconvincing (Liebowitz 2003). For instance, while CD unit sales were down by 10% in 2001, the figure cited by the RIAA, total revenue was down by 2%.¹⁰ Furthermore sales of recorded music are declining from an all-time high, and have experienced significant drop-offs in four periods prior to the widespread use of the Internet for exchanging digital music. The current slump began in 1999 when file-sharing networks had very little market penetration. Liebowitz discusses many factors which could contribute to a drop in sales for consumer entertainment goods such as the CD and concludes that, based on the figures, it is far from conclusive that the music industry is in terminal decline.

Recent studies by Forrester Research would suggest that usage of file-sharing networks affects different segments of the CD buying public in different ways. Heavy CD buyers tend to buy even more CDs, using the file-sharing Networks to try out new music before buying. Light CD buyers may be buying less, tending to burn their own compilations to use in portable walkmans or at work.¹¹

The key issue is that CD sales are declining and the music industry believes that it is due to the growth of on-line piracy. At the same time consumers are downloading and swapping music as much as ever, and do not feel that they are breaking the law. The type of litigation being pursued by the music industry in the US is to criminalise and alienate those people (16–30 year olds) who typically buy a lot of CDs. There appears to be a breakdown of the expectations between music providers and music consumers. The question is whether creators, producers, distributors and consumers can come to an agreement where each receives the value they expect.

The central issue in this debate is *copyright*, which, particularly as applied the music industry, is complex and difficult to understand (Fisher 2003). As we shall see, the music industry uses recording copyright as the single means of leveraging control over the physical production and

⁸ Recording Industry Announces 2001 Year-End Shipments. February 25, 2002

<http://www.riaa.com/news/newsletter/022502.asp>

⁹ RIAA on piracy <http://www.riaa.com/issues/piracy/default.asp>

¹⁰ <http://www.riaa.com/news/newsletter/022502.asp>

¹¹ Forrester Research Report. Bernoff, J. (2002) Downloads Save The Music Business

<http://www.forrester.com/ER/Research/Report/Summary/0,1338,14854,00.html>

'Proof' - downloading music hurts Europe's CD sales

<http://www.europemedia.net/shownews.asp?ArticleID=14535>

distribution of music. For them it is crucial that they leverage the same degree of control over the digital distribution channel.

Ironically, the original appeal of digital music to the industry was that – now that there was no physical product to be manufactured and distributed – they could ‘dis-intermediate’ distributors and retailers and market directly to the consumer, thus preserving the currently centralised distribution model minus a few value links. However, the lack of agreement on standards for digital authentication, digital distribution methods and payment models meant that the recording industry was caught unprepared by the advent of file-sharing networks. We suggest that the file-sharing networks not only represent a consumer desire for greater access to music but also a change in the way music should be marketed and distributed. We suggest that the success of the file-sharing networks is partly due to the alternative manner in which music was distributed – not from a centralised server, but from a community who in a self-organising manner have assembled the biggest repository of music yet available.

We suggest that the music labels, rather than attempting to establish direct marketing links with their customers, should adopt the role of an enabler or facilitator allowing users to share music with each other, thus facilitating a social network rather than simply a business relationship. It is clear that the record industry must be remunerated for digital services, but we suggest that the current pay-per-download model will only be attractive to certain segments of the market. We propose Smart Radio as an alternative model that offers a compromise service which caters for needs of the music industry and its consumers.

The remainder of this section will discuss the issues of digital distribution. We will first look at the competition the music industry faces in the form of file-sharing networks. We will then look at the physical distribution model, and the new digital distribution model. A sub-text running through this chapter is that the file-sharing issue has simply brought the long-standing problems within the music industry to crisis point. These include:

- Innate distrust by consumers of the music industry.
- The music industry’s high profit margin on each CD, which is used to ‘subsidise’ non-profitable artists. This argument is hard to make in a digital environment.
- The concept of copyright ‘ownership’ on a medium (music) which is simply not understood by consumers.
- The domination of the music market by five media conglomerates (the 5 ‘majors’).

2.1.1 What is a File-sharing Network/P2P Network?

The network architecture we are most familiar with today is the client-server architecture in which a *centralised* server hosts resources which can be requested by any number of clients (see Figure 2.1). The resources might be web page hosted by a HTTP server, or any type of file hosted by an FTP server. The server provides a single point of access and, in the context we are examining, if it hosts illegal material, it can be shutdown.

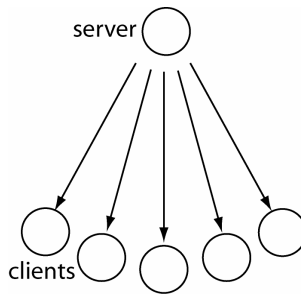


Figure 2.1: The client-server architecture

A Peer-to-Peer (P2P) network is a *decentralised* network in which each node can equally act as a server or a client (see Figure 2.2). In ‘pure’ P2P networks all peers communicate symmetrically and have equal roles. However, most P2P networks are not completely decentralised. Most have some degree of centralised functioning such as when a new host is bootstrapped. The Napster network was an example of a hybrid P2P system. While file-sharing was decentralised, the directory of files was centralised, with the Napster servers replying to search queries and brokering client connections (Oram 2001). It was the centralised nature of the Napster network architecture which allowed the US federal appeal court to rule that Napster was knowingly assisting the breach of copyright. Since the demise of Napster, many alternative P2P networks have sprung up which decentralise the search and brokering requirements of the network. The key point in file-sharing networks is that there is no single point of access. In theory, clients can download files from each other without going through an intermediary server. This type of service is very difficult to police.

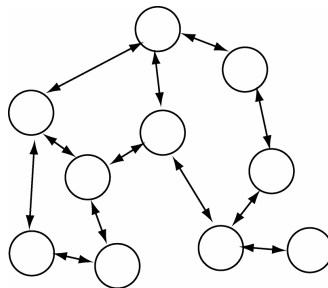


Figure 2.2: The figure illustrates the decentralised architecture of a P2P network

Decentralised systems are not new; the Internet routing architecture itself is largely decentralised, with the Border Gateway Protocol used to coordinate the peering links between various autonomous systems.

2.1.2 The MP3

The process of converting a sound into a digital format for transmission or storage purposes produces very large file sizes. For instance 1 second of what we term ‘CD quality’ sound requires a file size of 176.4 KB and the three and half minute pop single requires a file size of 37 MB. The mp3 was developed as part of an ISO initiative to develop standards for the encoding of audio and video files. The Moving Picture Experts Group (MPEG) was constituted for this task and met for

the first time in May of 1988. The main application the group had foreseen for the first audio-visual standard, MPEG-1, was CD-i, an interactive compact disc developed by Philips and Sony to put games and educational programmes on television sets. As part of the development of the standard, the expert group sought proposals for an audio codec (coder-decoder), a means of shrinking audio file sizes without losing its identifying qualities. The codec that was chosen had been developed in the 1980s by researchers from the Fraunhofer Institute and the University of Erlangen, Germany in order to allow high-quality music to be transmitted over ordinary telephone lines. The codec could shrink music files by a factor of twelve or more with little loss of quality. The MPEG-1 standard was completed in 1992; it described three separate but related ‘layers’, schemes for converting sound into a digital format. Layer 1 and Layer 2 were intended for high-performance applications; Layer 3, a version of the codec developed by the German team, was intended for devices that handle data relatively slowly, such as personal computers. MPEG-1, Layer 3 is what is now called mp3.

The mp3 codec is *lossy* – its compression algorithm removes a certain amount of data that cannot be recovered when the file is decompressed. The algorithm takes advantage of a feature of human auditory perception, called auditory masking, whereby the ear cannot discern certain frequencies in particular situations. For instance, this occurs when the presence of a strong audio signal makes weaker audio signals in the proximity imperceptible. The mp3 compression algorithm removes the tones people do not perceive, decreasing the size of music files without greatly affecting the sound.

The source code for the mp3 compression algorithm was stored on an insecure computer at the University of Erlangen where it was downloaded by a hacker, *soloH*, who produced software that could convert CD music tracks into compressed digital files, a process known as *ripping*. The software was further improved by a community of on-line developers. Within 2 years mp3 sites began to spring up throughout the Internet, all containing copyrighted songs that had previously been imprisoned within compact discs (Mann 2000).

2.1.3 Techniques of Music Delivery

There are two major methods of delivering audio and video content over the Web. The first method uses a standard Web server to deliver the audio and video data to a media player. The second method uses a separate streaming media server specialised to the audio/video streaming task. Until recently, audio and video on the Web was primarily a download-and-play technology. The entire media file had to be downloaded from a server before it could be played. However, this technique causes significant delays before playback because media files are usually large and take a long time to download.

The alternative to waiting for a full download is to *stream* the media files so that playback can begin while the data is being sent, without having to wait for the whole file to download. Playback begins after the client has pre-fetched a few seconds of the media into a buffer, after

which time it begins to drain the buffer and playback begins. As we can see from Figure 2.3, if the rate at which the client drains the buffer is faster than the fill rate from the network, then playback will have to pause while the buffer is refilled. However, increased broadband access has meant that streaming audio is an increasingly viable option for home users.

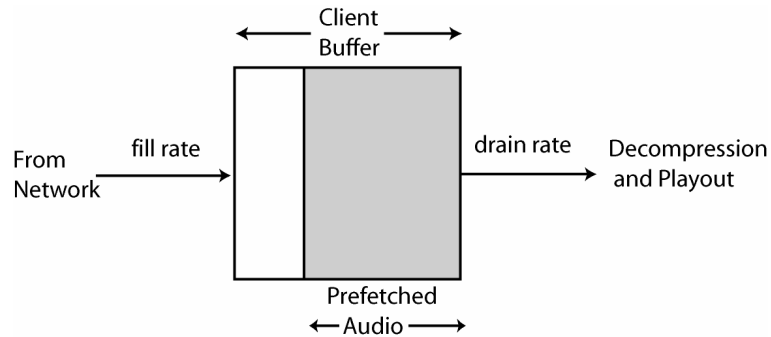


Figure 2.3: Client-side buffer for a streaming application

2.2 Physical distribution

The music industry uses a centralised model of distribution that has been built up over many years. It is quite a simple model whereby the record label controls the recording, sales and marketing and very often distribution of CDs. Smaller labels will license independent distributors to get their product to wholesalers, and thus incur a higher distribution overhead.

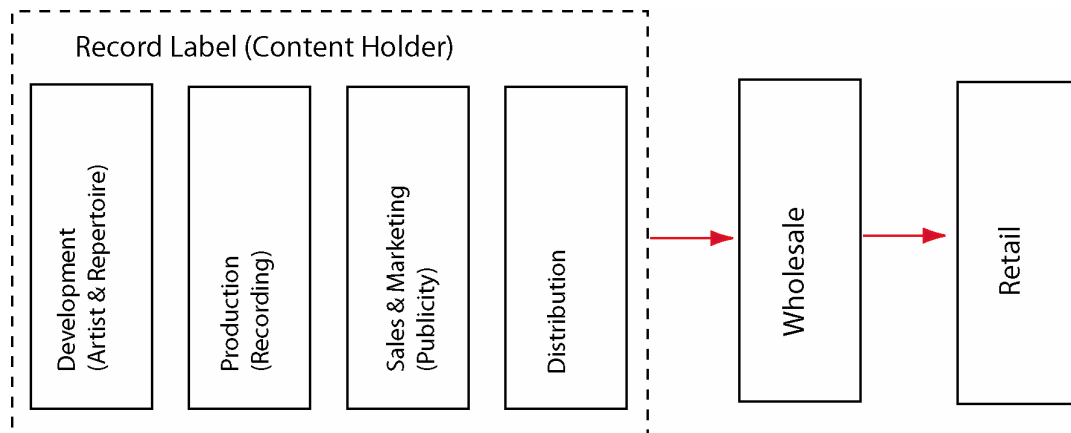


Figure 2.4: The music value chain for off-line distribution (Durlacher 2001)

The business model of the recording label is dependent upon leveraging the intellectual property copyright of the artist's recording. In the UK, the 1988 Copyright Act states that the copyright owner of a sound recording is the person 'who made arrangements for the recording to be made'.¹² This is taken to mean the person who pays for the recording. As a standard practice labels demand that musicians give up the copyright on the recording (Harrison 2001). This is contrary to the practice in the publishing industry, for instance, where authors own their books but license them to

¹² UK Copyright Designs and Patents Act 1988.

publishers. Thus most CDs bear a stamp displaying the label's ownership of the recorded music such as "copyright 2003 BMG". The argument put forth by the record label is that the ownership of the copyright allows them to invest in the artist by providing the services illustrated in Figure 2.4: Artist Development, Recording and Production services, Sales and Marketing and Distribution.

However, even after an artist has recouped with profit the label's outlay, the copyright still belongs to the label. Artists and their managers view the practice of insisting upon outright ownership of copyright as an abuse of record company power (MMF 2001). The artist's cut from the sale of the CD is low – 18% of the dealer price would constitute a good deal. However, the royalty calculation invariably contains several 'standard' reductions for expenditure such as packaging and manufacturing (20–25%) and the producer's royalty (3–4%) so the artist receives considerably less than suggested by this figure. On top of this, the cost of recording and any advances the artist receives while recording are recoupable from the artist's royalty rate. Recording royalties are not paid by the record company until such costs have been recouped. Therefore, the artists recording royalties are dictated by how many units are sold. In many cases they make very little or nothing from recording royalties after the recording label's costs have been recovered (Mann 2000). Instead, if they write their own songs they will make money from mechanical royalties, which are technically paid for every song on every CD. In the UK the Copyright Tribunal 1992 has set this rate at 8.5% per CD. This royalty is paid from the record company's cut of the CD, irrespective of the number of sales that take place, and is generally the songwriter's biggest source of income (Harrison 2001).

Figure 2.5 shows the price breakdown of a new CD (€21.45) in Ireland. The figures are based on a cost analysis from the MMF guide to professional music management (MMF 2001). The figures are based on the assumption that the record company is a major with its own distribution network, thus distribution costs are relatively low. It must also be kept in mind that the artist royalty rate (5%) is only paid once the record company has recouped recording costs and artist advances. Until the CD has reached this quota the artist will not receive recording royalties.

According to this analysis, the record company takes five times as much as the artist. However, after taking into account the marketing and promotional expenditure the profit ratio might typically be in the region of 2:1, or even nearer 3:1 in favour of the record company (MMF 2001). The recording industry argues that these ratios are necessary to cover the expense of the number of artists they lose money on – according to one report 90% of the roughly 20,000 albums released every year sell less than 10,000 copies.¹³ While the number of copies necessary to cover costs depends on the investment, one analysis suggests that the company needs to sell 86,957 CDs to cover costs on an average first album deal (Leach & Henslee 2001). In fact only 16% of all record releases reach that sales figure. According to the same analysis it is typical for 5–10% of the labels' roster of artists to subsidise all the music released by the label (Leach & Henslee 2001).

¹³ International Data Corporation cited in Hartmann (2000).

Record labels argue that lowering profit margins will mean that they will be forced to take less risk, thus reducing the number and diversity of artists that they sign.¹⁴

This is a double-edged sword. Non-profitable artists *do not* receive recording royalties from a label because costs are recouped from the artists share until profitability is reached. Therefore the RIAA claim that piracy hurts artists, refers to the profitable 15% of artists who still only receive a small portion of the price per CD. Of course, piracy also means that songwriters do not earn from the mechanical copyright of the song. However, this area is not within the remit of the RIAA. In fact, the recording industry has regularly fought to reduce the artist's mechanical copyright fees. The point of this analysis is that the smaller artist has much to gain if another mechanism for getting his/her music to the public outside the conventional means of production and distribution can be established. On the other hand, the music industry has a lot to lose if control of production and distribution is decentralised. As Frith (1993) has pointed out in his analysis of music copyright, music industry concern at the plight of the artist is usually an alibi for protecting its own interests.

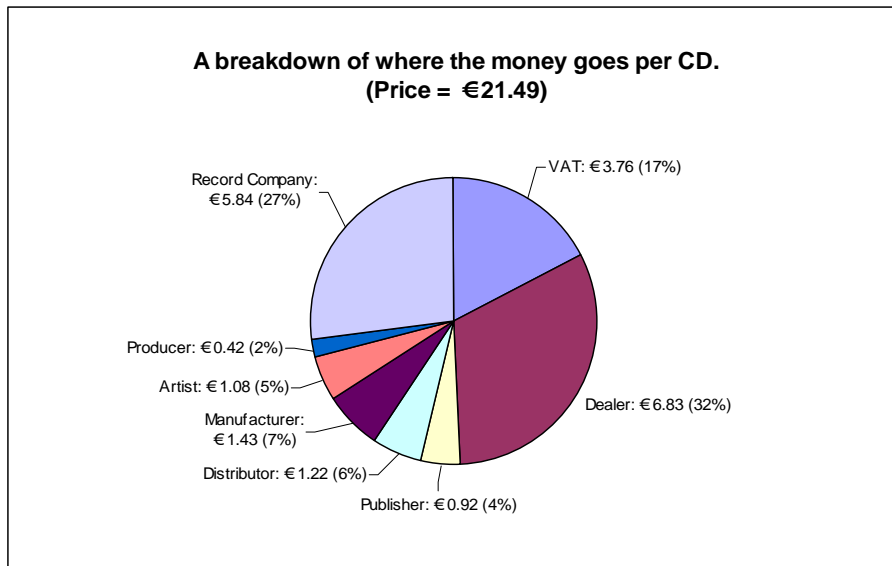


Figure 2.5: The breakdown of the price of a CD

2.3 On-line Distribution

On the surface, on-line distribution appears to be a simpler, more direct model and offers benefits to the music industry and consumers. According to Fisher (2000) the potential advantages of digital distribution for the music industry are:

1. Cost savings due to 'disintermediation', the removing of aspects of the physical supply chain and developing a direct relationship with the consumer.

¹⁴ RIAA on anti-piracy <http://www.riaa.com/issues/piracy/default.asp>

2. Elimination of overproduction and underproduction. Record companies would no longer have to guess how many copies of a CD they should manufacture.

The potential advantages to the consumer are:

1. Immediacy and precision: consumers would no longer have to wait for a CD to be in stock, or for a mail order purchase to arrive. Digital distribution offers the opportunity of buying only those tracks you like, and not a whole CD.
2. Increase in the number of and variety of musicians. Musicians that appeal to ‘niche’ markets would find it easier to distribute on-line than through the CD medium.
3. Semiotic democracy: Fisher argues that the power to shape culture is often concentrated in relatively few hands. In relation to the previous point, he suggests that digital distribution has the potential to decentralise the semiotic power of the major music labels as more artists than ever before can be made available. Secondly he suggests that the ease with which “consumers” can manipulate a digital asset, recombine and blend it with their own material will allow them to become “producers”.

One thing that is striking about Fisher’s analysis is that the advantages for the consumer seem to be gained at the expense of the recording industry. In a later essay, Fisher points out that intellectual copyright rules are difficult for the consumer to understand (Fisher 2003). This is problematic for the music industry which relies upon recording copyright as its single means of leveraging value. Consumers may be resistant to the charging systems in place for digital distribution when there is no physical product being purchased. People are used to purchasing music as a tangible product with packaging extras such as cover art. When the physical product is removed, although this is a small part of the overall cost, the music industry faces the dilemma of convincing consumers of the value of their digital product. From a consumer perspective the charging systems in place for digital download may seem exorbitant and poor value for money. The industry’s case is not helped by general consumer suspicion that they have been overcharged for CDs for years. This, in fact, has been proven to be the case. In January 2003, a US district court ruled against the major labels in a lawsuit brought by the attorney generals of 43 US states alleging price fixing during the period 1995 to 2000 (Compact Disc Minimum Advertised Price Antitrust Litigation Settlement).¹⁵ The recording industry chose to settle rather than appeal this ruling.

2.3.1 On-line Distribution Model

Translating the physical distribution model into an on-line distribution model has not been an easy task for the music industry. The digital value chain has turned out to be longer than the real world chain described earlier (Birch & Davidson 2002). We suggest that this is because the mechanisms required to protect copyright in a digital environment are complex. Figure 2.6 illustrates the value chain for digital distribution which is adapted from a 2001 report on the economics of digital

¹⁵ <http://webform.musiccdsettlement.com/english/>

distribution of music (Birch & Davidson 2001). While the value chain suggests roles for many different technology and service providers, the pressure to compete with the file-sharing networks has driven large entertainment and technology companies together to provide solutions that are, in many cases, as centralised as the physical distribution models. We will use Figure 2.6 as the basis of our analysis of on-line distribution because it succinctly represents the tasks required for on-line distribution, irrespective of whether these are handled by the same media conglomerate.

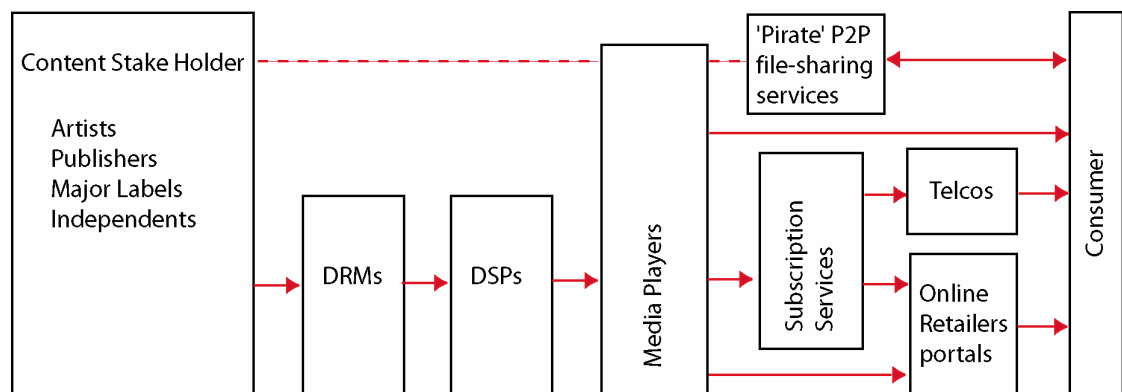


Figure 2.6: The figure illustrates the value chain for digital music distribution (OC&C 2001)

2.3.2 Content Stake Holder

Artist/Songwriter: Artists are increasingly aware of the power of the Internet as a means of getting their music to an audience. Many artists, aggrieved at what they perceive as the monopoly by the records labels over physical forms of distribution, are not persuaded by the record industry's attempt to maintain the same profit margin when there is no physical product to manufacture and distribute. This has resulted in artists sending cease and desist notices to their labels in an attempt to renegotiate their contracts for distribution on-line. This has resulted in limited on-line availability of the music of certain artists.¹⁶ In fact, many artists are beginning to wonder whether they need a record label if they can sell on-line. They argue that higher margins more than compensate for lower sales due to reduced promotional support.

Record Labels: Originally record labels viewed digital distribution as a means of cutting out the current distribution and retail overhead and capturing their margins. Generally, labels are realising that their strength lies in content provision not in technology development or retailing. However, their central concern is to protect the integrity of their core asset – content. Thus, as long as digital sales remain a small part of their overall income, they will support any model which strengthens the current off-line model, price points and margin for physical product.¹⁷ Until recently, many of

¹⁶ Helmore, E. (2001) Guardian, December 13.

¹⁷ Birch, A., Davidson, S. (2001) Digital dilemma – making music, losing money. OC&C report.

the current on-line retail models offered content at a price which is directly comparable to the CD price.¹⁸

Digital Rights Management (DRM):

Digital Rights Management refers to controlling and managing rights to digital intellectual property. As we discussed in Section 2.1.2, the mp3 format was designed to reduce the size of audio files. Like any digital artefact, multiple identical copies of an mp3 can be made and distributed outside the control of the copyright owner. DRM technology seeks to maintain the authority of the original, by limiting the rights the user has to the copy. Stefik (1997a,b) specifies three types of digital rights:

1. **Rendering rights:** define the means by which the copy can be viewed, printed or in the case under discussion, played. It also relates to granting the user a time-limited period to use the copy.
2. **Transport rights:** define the user's permissions to copy, to move or to loan. In the debate over defining a secure mechanism for digital music, this refers to the user's right to make copies of music files, to use them on portable devices or to swap them with others.
3. **Derivative rights:** refer to the user's permissions to extract, edit and embed content from the copy in another digital work.

There are three ways to enforce content rights:

1. Through legal means using registration forms, license agreements, and copyright laws.
2. Through legal means by auditing watermarks (unique identifiers permanently embedded in the content).
3. Technologically, using encryption and user authentication to protect content and only make it accessible under strictly specified conditions.

Most DRM schemes for music nowadays use encryption. However, encryption is often not enough. A major issue in DRM management is the 'container' in which the digital content is rendered. If the container allows the user to make unauthorised copies of the digital content then it is 'untrusted'. A 'trusted system' (Stefik, 1997a,b) is a system in which the hardware and software can be relied on to follow certain rules pertaining to the usage rights for the digital content. An example of DRM is the content scrambling system employed on DVD disks which is encrypted so that it can only be decoded and viewed using an encryption key, which the DVD Consortium keeps secret. The DVD player manufacturer must agree to sign a licence agreement with the DVD Consortium which restricts them from including certain features in their players such as a digital output which could be used to extract a high-quality digital copy of the movie. However, the DVD encryption has already been hacked. The decryption software has been bundled with a compression

¹⁸ Helmore, E. (2003) Apple tunes in, but music still has a problem. Observer May 4;
<http://observer.guardian.co.uk/business/story/0,6903,948880,00.html>

format called DivX which allows pc users to rip and store DVD movies on conventional CD ROMs¹⁹.

Currently, the entertainment industry is pressing for changes to the hardware and software architectures of current computers so that they can be used as trusted systems.²⁰ A trusted computer, for instance, would refuse to make unauthorised copies or to play audio or video selections for a user who has not paid for them. However, this conflicts with the interests of consumers for whom the computer is an *all purpose* device for the manipulation of information whether for work or entertainment purposes. Despite this Microsoft has announced integrated DRM features in a forthcoming secure windows environment, Palladium, which could curb software piracy as well as music piracy.²¹

Despite attempts by the RIAA to establish a standard DRM for music,²² two proprietary, non-compatible standard technologies have recently been adopted by Digital Services Providers on the web: Microsoft's Windows Media DRM²³ and RealNetworks DRM.²⁴ These technologies can impose limitations on which platform the user can play the music file, how long the user may use a downloaded file and whether it can be used on a mobile device. One way to do this is to encrypt the file and register it with a database. Every time the file is used the on-line database is checked for authorisation. If this is not received the file access fails. Since DRM systems are continually hacked, multiple licence updates are required.

Digital Service Providers (DSPs):

The role of the Digital Service Provider is to license its music distribution platform to companies seeking to sell music services under their own brand. They function both as a technology provider and a music clearing house. The best known players in the US market, *MusicNet*, *PressPlay*, and *Rhapsod*, are ventures controlled by the large record companies and Network service companies. MusicNet was founded as a coalition between RealNetworks, AOL-Time Warner, Bertelsmann and EMI, while PressPlay is a partnership between Vivendi-Universal and Sony. The Rhapsody service is distributed by Listen.com, an independently founded company which has recently (April 2003) been acquired by RealNetworks. With so few players in the market, attention has been drawn to the difficulty in securing digital distribution rights for independent DSPs, particularly where there is a conflict of interests with the DSP ventures of the major labels. Anti-trust investigations have been set up by the European Commission, and by the US Department of Justice to investigate whether

¹⁹DivX website. <http://www.divx.com/>

²⁰ A Bad, Sad Hollywood Ending. Business Week online 16/06/2002.

²¹ Can we trust Microsoft's Palladium? Salon.com. July 11, 2002.
<http://www.salon.com/tech/feature/2002/07/11/palladium/index.html>

²² <http://www.sdmi.org/>

²³ <http://www.microsoft.com/windows/windowsmedia/press/prdrm.aspx>

²⁴ <http://www.realnetworks.com/products/drm/index.html>

the majors protect their own ventures by offering non-competitive licensing agreements to independents.²⁵

The *iTunes* service from Apple computers does not follow the same model as the other providers in that it sells directly to users of Apple computers. Apple's incentive was to provide a service for MAC owners who had generally been overlooked by the paid services and the file-sharing services. A Windows iTunes service is scheduled to operate before the end of 2003. Spurred on by Apple's entry into the market place, Microsoft has recently (August 2003) joined with a London-based DSP to provide the first pan-European service selling songs on a pay-as-you-go basis. However, EU anti-trust regulators state that the move bolsters their case against Microsoft for squeezing out competition in the media player market.²⁶

In section 2.3.3, we will briefly examine the services operated by three of the best know DSPs. However, this section will have made clear that the collective panic of the majors in the face of file-sharing has led to the operation of monopolies in the marketplace. This has been exacerbated by a series of mergers and acquisitions which have made it difficult to distinguish between content-provider, distributor and technology-provider.

Media players:

A standard DRM architecture for music would have allowed several music media player vendors into the market place. As it stands, the proprietary DRM mechanisms of RealNetworks and Microsoft are designed to be used with their respective media players: Real Player and Windows Media Player. This squeezes out the numerous other media players such as WinAmp which play unsecured media formats. In some cases, such as Rhapsody, the player software is built into a new (DRM protected) player device.

However, as we stated earlier, recent mergers have abolished the distinction between Media Player vendor, DRM vendor, content provider and service provider. Furthermore, the bundling of the Microsoft Windows Media player into the Windows Operating System means that the Microsoft player is unfairly poised to take advantage of the growth in legally regulated services. Although this is now the subject of an EU Commission anti-trust investigation, it remains to be seen what effective action, if any, can be taken.²⁷

2.3.3 Services on Offer

Table A-1 in Appendix A summarises the services offered by three of the major DSPs mentioned already. The key issue is the type of service they offer the consumer in comparison to the file-sharing networks. In terms of GUI design and delivery reliability, paid services generally offer

²⁵ Duke, L. (2001) Tech. Rev. 0039 <http://www.law.duke.edu/journals/dltr/articles/2001dltr0039.html>

²⁶ <http://www.wired.com/news/business/0,1367,60028,00.html>

²⁷ Microsoft Faces EU Fines, Software Curbs (2003) Reuters; Aug. 6:

<http://www.reuters.com/newsArticle.jhtml?type=topNews&storyID=3233204>

better service to the user than the file-sharing networks. File-sharing networks use a single service model which involves searching and downloading on a track-by-track basis. Once the track is downloaded there are no limitations on its use. The mp3 can be transferred to multiple computers, mobile devices or can be burnt to any number of CDs. There is no time limit or expiry date on its use. All of the paid services, except Listen.com's Rhapsody services, have chosen to follow the download model. However, these music services use DRM techniques to protect the music files they sell to their customers. In most cases this means a restriction upon whether the music file can be transferred to another computer or mobile device or burned to a CD, and limitations on how long it can be used. This poses a serious drawback for consumers used to downloading mp3s from file-sharing networks for unrestricted use. Another disadvantage is that the catalogue of music being offered by paid services is considerably smaller than that available from file-sharing networks. In fact, the current on-line models offer little to the user other than the benefit of being able to pick and choose tracks at will. Indeed, the service models currently on offer emulate the centralised physical distribution model, with few advantages for the consumer over directly purchasing a CD in the shop. In many respects digital distribution has offered the music industry the perfect sales opportunity – a direct relationship with the consumer, selling time- and location-restricted items, which unlike CDs are difficult to counterfeit. The consumer appears to be the loser in this equation.

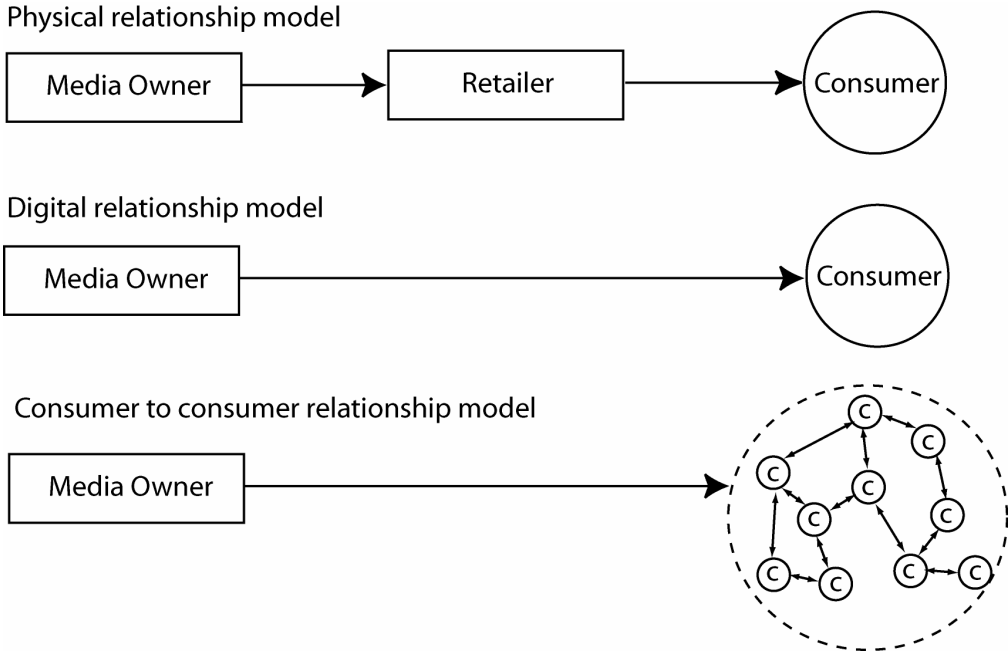


Figure 2.7: Three relationship models

Figure 2.7 illustrates three different relationship models. The first models demonstrate the current physical distribution model where consumers buy directly from retail stores. The second model is the current digital version of this, in which media and network conglomerates are able to sell directly to the consumer. The third model offers an alternative, in which the distributed nature of

the Internet is fully realised. In this scenario, media companies take the role of content providers, with community driven services enabling person-to-person services.

We will discuss later in this chapter how the distribution of music in particular is suited to this particular framework. In section 2.6 we will introduce Smart Radio, a community-based, streaming playlist service as a prototype of this model. As such we view this as a compromise solution between the current paid services and peer-to-peer services.

PressPlay:

PressPlay offers a subscription model where the user has access to unlimited downloads and unlimited streams. Microsoft media DRM is used to control the files delivered to the user. Downloaded files can be stored on up to two computers and are enabled as long as the user's subscription is up-to-date. If the user wishes to move downloaded files on to a mobile device or burn them to a CD, then he/she has to pay a 'portable' download surcharge of approximately \$1.00 per song. (The exact price is dependent on the 'pack' option the user chooses. See Table A-1, Appendix A). In addition, the mobile device must support the secure Windows Media file format (WMA) or Net MD™, the Sony secure mobile device format. The streaming service allows users to stream entire albums, or playlists of songs.

Recommendation service:

Users can search by artist or by genre. As well as the results from the search, a selection of other artists that the user may like is presented based on the download and streaming behaviour of other users. The user can also manually search to see what other users are streaming. The 'Build your Own Station' facility allows playlists to be generated on the fly based on recommendations from the tracks the user has downloaded. The playlists are divided by genre.

iTunes

Apple's iTunes music store is the most recent service that offers downloaded music from the big 5 labels. It is a pay-as-you-go download service that operates from Apple's iTunes client player, and unlike PressPlay, it does not require a subscription. Users simply pay and go (\$0.99 per track). Apple claims that the service gives users "the same level of rights as people who buy CDs".²⁸ While this claim is exaggerated, the service is markedly less restrictive than that offered by Press Play. Apple uses a proprietary DRM in conjunction with the ACC compression format (ACC forms the basis of the MPEG-4 audio compression technology²⁹).

Playlists containing any protected AACs can only be burned 10 times. The list must be manually changed before it can be burned again. AACs can be transferred to up to three computers and any number of iPods, Apple's mobile device.

Recommendation service:

²⁸ <http://news.bbc.co.uk/2/hi/entertainment/3148327.stm>

²⁹ <http://www.vialicensing.com/products/mpeg4aac/standard.html>

When a user previews a song he is presented with a set of songs that other users have purchased in addition to this song. New playlists can be automatically generated from songs the user has already downloaded using the 'Smart playlists' facility whereby the user can specify criteria such as genre or artist to be included in the playlist.

Rhapsody

Rhapsody offers a subscription *streaming* service where users choose songs or playlists of songs to stream to their desktop. Unlike PressPlay or iTunes, Rhapsody does not yet offer the facility to download and move files to a portable device. However, songs may be burned to a CD for a surcharge of \$0.99 per track. Since there are no downloads, users can use the service from any computer they choose. Playlists can be mailed to a friend. However, there is no automated way of identifying users who have a similar taste.

Recommendation facility: The service offers a facility whereby expert 'editors' create a new playlist suited to the user's taste in music. For scalability reasons this must be at least a semi-automated task. User can also choose a particular artist as the basis for a particular stream of music. Rhapsody will stream a mixture of items by the chosen artist and artists similar to this artist.

2.3.4 Discussion

Paid music services are in their infancy, and it is unclear which models are attractive to consumers. DSPs are unwilling at this stage to disclose sales figures. It is clear, however, that the Rhapsody service is relatively successful having signed syndicated deals with several retail portals, and consistently receiving good press for its interface and ease of use. MusicNet, a subscription download service like PressPlay, appears to have lost ground to Rhapsody. One of its main outlets (and a stake holder), Real Network's RealOne, has recently committed to the Rhapsody service model. iTunes opened its account with a fanfare in May 2003. According to press reports, opening day downloads equalled the number of songs legally downloaded over a six-month period in the previous year.³⁰ Apple's less restrictive DRM and *pay-as-you-go* service may be important factors here. It also may simply be the fact that they are servicing a captive audience. Up to this point both file-sharing networks and pay services have only catered for Windows users. Some commentators have noted that Apple's more liberal DRM policy may reflect the more affluent demographic of the average MAC user who theoretically would not be inclined to engage in hacking the ACC audio files for posting to windows based file-sharing networks.

It is unclear whether the pay models offer sufficient advantages to the consumer to make a dent in the file-sharing 'market'. While download reliability and ease of use are beneficial, restrictive DRM clauses present major drawbacks. Of the three models we present, one (PressPlay)

³⁰ Kahney, L. (2003) Music biz buzzing over iTunes. Wired News. May
<http://www.wired.com/news/print/0,1294,58760,00.html>

has chosen to compete directly with the file-sharing networks providing unlimited downloads at a subscription price of \$10 per month. Since downloads expire when a subscription ends, unless purchased with an additional surcharge, we would suggest that such a service would have great difficulty in attracting users who currently use the file-sharing networks.

iTunes, although a download service, is initially targeted at providing a service for Mac users to stop any leakage of their customer base to the Windows platform, where the pay and non-pay services can be found. iTunes is due to release a Windows version later in 2003, which will be a greater test of their sales strategy.

Rhapsody, on the other hand, has chosen to offer an alternative service to file-sharing networks. The service is akin to a piped music service or a radio service. The service has been in operation since April 2001 and appears to have prospered. It recently was acquired by RealNetworks, despite Real's stake hold in the MusicNet (download) service, and will be incorporated into the RealOne service which is available from every Real Player.³¹

2.4 Value-added Services

We would suggest that paid services could attract users by offering facilities not available up to now from the file-sharing networks. Paid services are intrinsically centralised models in which the service provider is in a position to *learn* from the preference data expressed, either explicitly or implicitly, by users as they use the service.

By acting on such preferences the service provider is in a position to develop novel means of distributing music. For instance we would suggest that the service provider could address the problem of music overload not by providing anonymous recommendations, which the consumer may not trust, but by enabling communities where users explicitly make recommendations to each other.

2.4.1 Personalisation

The advent of on-line music services poses similar problems of *information overload* often described for textual material (Foltz & Dumais 1992, Resnick & Varian 1997). It is often acknowledged that the volume of live and archival information available to the user through the Internet has made it difficult to locate relevant information in a timely fashion. Search engines may retrieve too many results for the user to sift through. One solution that has been gaining broad acceptance is the concept of *personalisation* which refers to the task of customizing information access, retrieval handling and display according to the user's preferences (Mobasher et al. 2001). Personalisation systems generally have three broad steps (Mobasher et al. 2001, Goker 2002):

1. User preference data is acquired through the user's interaction with the service.

³¹ Mook, N. (2003) Real Dumps MusicNet for Rhapsody. Beta News May 28, <http://www.betanews.com/article.php3?sid=1054116913>

2. The data is analysed in order to build a user model.
3. The user model is used as a basis to automatically adapt and generate customised information or behaviour.

The advantage of the personalisation process for the user is the reduction in work due to the automated presentation of targeted information based on the user's long-term/short-term interests.

The advantage for the service provider is that customer *loyalty* is fostered. Cutler and Sterne (2000) define loyal customers as the key to an on-line service since they “come back frequently, buy often, recommend your company to others and readily try out new things”.

Personalisation is clearly going to be an important aspect of the on-line service for music. As we can see in Table A-1 in Appendix A, the large service providers offer 100s of thousands of music items. It is estimated that this represents less than 20% of the music that is currently available on CD. As digital licensing becomes commonplace, service providers will be hosting millions of tracks, and will need to implement personalised search facilities that takes into account the taste of the user and how the user listens to music. Unlike perusing a result set from a Google query where each page can be quickly scanned, audio files must be first (partially) downloaded and previewed in real time, requiring a much greater investment by the user.

2.5 Search and Personalisation

In this section we examine personalisation in the domain of multimedia, particularly music. To examine the issues we briefly look at how the search is conducted in the domain of text retrieval where content description is readily available. Figure 2.8 illustrates the main tasks carried out by an Internet Search Engine. Firstly, websites are ‘crawled’ using the web’s hyperlink structure. For each page encountered it extracts a list of words and associated values such as the position of the word on the page, the font, whether it is a heading or body text and its frequency in the page. The information extracted from each page is dependent on each search engine. In Google, for instance, each document is represented by a *hit list* which corresponds to a list of occurrences of a particular word in a particular document including position, font, and capitalization information (Brin & Page 1998). The words and associated pages are indexed using the search engine’s weighting system. Different search engines employ different heuristics for weighting the features extracted from each document. Users can then search for documents by submitting words or combinations of words. The engine searches its index for these words and retrieves a ranked set of documents.

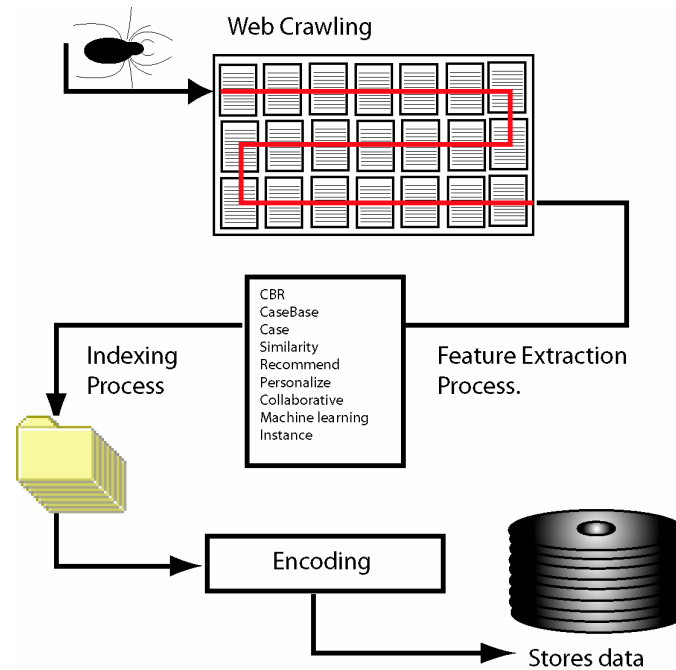


Figure 2.8: The feature extraction and indexing process used for Internet search engines

The key point is that it is relatively easy to parse meaningful features from a text document and thus index the document for retrieval.

2.5.1 Personalisation

We can view some aspects of personalisation as a type of automated search strategy where the goal is to anticipate the user's future requests based on an analysis of his/her previous behaviour. We therefore require content descriptors to construct a user profile, which is a representation of a user's interests. User profiles can capture long-term or short-term interests, and are generated by analysing how the user makes use of the system resources. We discuss this subject in more detail in later chapters.

One means of generating a user profile is to analyse the content being viewed by the user and to extract key features which are then used to represent the user's interests (see Figure 2.9). For instance, in the Fab system, Balabanovic & Shoham (1997) represent the user's interest as a vector of words extracted from web pages the user has viewed. The user's potential interest in a particular page is indicated by the similarity between the user profile vector and the vector of words representing the page. Similarity is calculated using the cosine rule (see Equation 4.4).

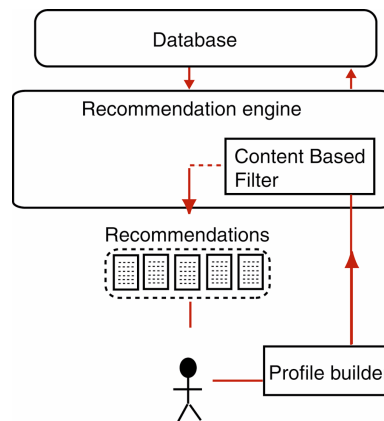


Figure 2.9: Personalisation requires an analysis of the user’s behaviour in order to anticipate future possible requests

2.5.2 Content-Based Music Retrieval

Content-based music retrieval refers to retrieval systems where music is retrieved on the basis of a description of the content of the music, in a similar way to the text-based retrieval we have just looked at.

While words, the key semantic elements, of a text document are readily available within a document, it is a non-trivial task to extract features from an audio artefact (Foote 1999).

There are two ways of assigning descriptors to audio content:

1. Feature extraction using signal analysis
2. Content mark-up using human analysis

2.5.3 Automated Signal Analysis

Audio retrieval using features from automated signal analysis has many techniques in common with image and video retrieval (Zhang et al. 1995). However, audio classification is defined by features of the human auditory system, and by the aural features which human beings consider meaningful. A wide variety of disciplines are involved in research on music retrieval such as signal processing, psychoacoustics, speech recognition, machine learning, semiotics, psychology and musicology.

In the case of automated feature extraction systems, Smoliar et al. (1996) suggest that there are two approaches to content-based retrieval of multimedia artefacts: “expression-based retrieval” and “semantic-based retrieval”.

Expression-Based Retrieval

In expression-based retrieval music artefacts are retrieved using a physical description, such as the frequency or amplitude of the waveform. Generally this is carried out using a Fourier analysis of the waveforms of the sound. With this technique the indices used are mean, variance, zero-

crossings, auto-correlation, and histograms of samples/differences of samples. These metrics can be taken over the whole sample or on blocks of data. The description is generally of a low level, with features that are difficult, if not impossible, to interpret without expert intervention. Some researchers have focused on indexing audio using these features and neural nets (Feiten & Gunzel 1994). Generally, a *query by instance* technique rather than a human level specification is used. However, the problem with this approach is that the neural net cannot (easily) provide an explanation as to what features are being used to determine similarity.

The *psychoacoustic* approach attempts to model how the human auditory system extracts desired information from the physical information. Researchers in this area have derived higher-level acoustic semantics by testing psychoacoustic models on humans (Plomp 1976, Cook 1999). Thus the psychoacoustic approach to audio retrieval assumes an *a priori* model of how the physical audio data is structured. A seminal piece of work in this area is the Musciefish³² System (Wold et al. 1996) in which perceptual features, such as brightness, loudness, pitch, timbre and harmonicity, are used to index a sound file. A normalised Euclidean distance and the nearest neighbour (NN) rule are used to classify the query sound into one of the sound classes in the database. The developers of Musciefish suggest that the feature extraction process and representation is suited to retrieval of audio archives where the audio pieces are short such as the sound of gunshots, footsteps, glass-breaking or long homogeneous sounds such as the sound of rain or running water (SoundFisher 2001).

Longer or more heterogeneous audio pieces (like music) required detection of the transition between sounds (a segment), and a separate analysis of each segment (Wold et al. 1999). This means that analysis, representation and similarity calculation are considerably more complex. The Musciefish technology is currently being used, not for personalisation purposes, but to counter piracy. The company was acquired by Audible Magic Corporation in October 2000 and is used in content detection software which can assign a unique fingerprint to each audio piece based on its perceptual characteristics. This is used to register an audio piece as the “original” and to detect pirated copies.³³ The route taken by this company would suggest that extracted features are useful for the purposes of uniquely identifying a piece of audio, but not for the purposes of retrieval or classification where interpretability is important. Although the Musciefish system uses a Nearest Neighbour algorithm, the low-level feature representation, possibly over several segments of audio, would still be difficult to interpret. For example, Table 2.1 illustrates an example from Wold et al. (1996) of an analysis of male laughter.

³² <http://www.musciefish.com/>

³³ <http://www.audiblemagic.com/>

Table 2.1: Male laughter, duration: 2.12571 (Wold et al. 1996)

Property	Mean	Variance	Autocorrelation
Loudness	-54.4112	221.451	0.938929
Pitch	4.21221	0.151228	0.524042
Brightness	5.78007	0.0817046	0.690073
Bandwidth	0.272099	0.0169697	0.519198

Other approaches to audio retrieval classification include research into how statistical pattern analysis on low-level features can be used to predict *genre* or *instrumentation* (Tzanetakis et al. 2000, Grimaldi et al. 2003). These approaches attempt to bridge the gap between physical and psycho-acoustical features and features which have human level semantics attached to them.

Semantic-Based Retrieval

Smoliar & Wilcox (1997) suggest that the features used by the Muscelfish system are not semantically useful for audio retrieval in which human beings have to input query descriptions. Drawing upon their own work in image retrieval they propose a higher-level representation using concepts such as *context*, *form* and *content*. This representation requires *a priori* knowledge such as, for example, whether a sound is a jazz recording (context). In which case, it can be segmented on the basis of a particular style, for example, large combo versus small combo (form). Finally, the segments can be searched for particular acoustic motifs such as trumpet alternating with a saxophone (Smoliar & Wilcox 1997). The motivation for this research appears to be to increase the level of *interpretability* of audio representation. However, this comes with an increasing reliance on human input to develop models that correspond to how audio is perceived and *used* by people in the real world.

2.5.4 Human Analysis

Human analysis refers to any form of content mark-up that requires humans to input description. In the domain of music we observe that there are both *formal* and *informal* semantic systems to describe music.

Formal

Formal systems to do with music are based in the various disciplines such as music theory, music-cognition, ethnomusicology and psychology.

Music theory concerns the structure or syntax of music. Its subject matter extends from the formation of scales and chords to procedures for the distribution of pitches in time such as counterpoint and twelve tone or serial operations to principles of musical form (Kerman 1986).

Music cognition is a branch of cognitive science which attempts to model the psychological/cognitive mechanisms underlying musical thought processes. A general principle of cognition is that the brain abstracts recurrent patterns from the environment and encodes them in

the form of schematic representations as a basis for future classification and comprehension. Music cognition research attempts to model these representations. Desain et al. (1998) provide a review of computational modelling of music cognition, suggesting that this research is hampered by a lack of any standard way to integrate, compare or evaluate differing models.

While Musicology is generally concerned with the history of western high art music, the field of ethnomusicology is much broader, considering non-western music, folk music and popular music. Ethnomusicology studies are anthropological in nature, examining music in relation to the society in which it is created. Research in folklore has a long-standing tradition of lyric analysis that has been extended to the lyric in popular music. Frith (1988) and Griffith (2003) provide an overview of this approach. Griffith addresses the problem of analysing lyrics outside the context of the music and suggests that analysis needs to address performance and song structural qualities in order to have a full understanding of the lyric.

In the field of psychology, Leonard Meyer's seminal work on music interpretation elaborated a formal syntax of music by identifying musical elements like repetition (riffs, beats, verses refrains), delay (unresolved phrases, codas, stuttered delivery), and closure (refrains) as devices that may stimulate specific psychological feelings of anticipation and release in listeners (Meyer 1956). Keil (1994) has responded with a study that describes musical elements that stimulate *bodily* responses to improvised music. While we might view these as *semiotic* approaches, which attempt to qualify listeners' responses to cues in the musical 'text', other researchers view music as a part of the 'social text' in which musical structures reflect the socio-political context in which it is created and heard (Shepherd 1991). This approach views music as a cultural artefact in which meaning is shaped by the larger ideological debates such as class, race and gender.

Each of the disciplines mentioned could be considered to have made a contribution to explaining music and its relation to the human subject, but in ways that cannot often be represented outside the bounds of that discipline itself. Even if representational structures were readily available, the formalisation and integration of knowledge about any piece of music would be a labour-intensive process. While these disciplines attempt an objective analysis of music, informal more subjective descriptions of music abound in terms of music reviews and everyday conversation.

Informal

We use the term 'informal' not in a negative sense but to describe the forms of music knowledge which are not based as part of an academic or formal framework. This is 'word of mouth' knowledge that most people are capable of sharing. This type of information is usually subjective, expressing likes and dislikes, perhaps characterising music in non-musical terms. Given that most people can be encouraged to voice an opinion, unlike formal systems this form of knowledge is

widely available. Broadly, this knowledge is informed by the historical, social and cultural context in which an audio artefact is located and which contributes to the meaning and use value.

The key observation here is that a piece of music is not the sum of its structural parts. Since it exists (generally) as a cultural object as well as a physical object, it is also subject to less quantifiable human perceptions such as emotive power, aesthetic merit, social and cultural fashion and context-of-use.

Informal forms of knowledge share the following properties:

- Subjective aesthetic appraisals
- Informal use of language
 - Description of music in non-musical terms (mood, emotive qualities) or actions i.e. ('Music that is good to listen to when working' or 'music that is good to dance to').
 - Description of music in terms of other artists ('Radiohead are like U2').
- Determined by social relations, culture, fashion, age (See section below).

Cook (1998) argues that social debate, discussion and participation are essential to the process of creating meaning out of music. One aspect of this is how music enables community. Cooke suggests that western musicology has lacked a proper analysis of how music is used to establish cultural and group identity. As a counterpoint he examines the interactive, communal musical traditions found in South African culture.

However, popular music in western culture has been responsible for defining meaning in several subcultures. Hebdige (1991) examines how music contributed to the creation of urban subcultures like *Punk* in the 1970s. Rose (1994) studies *Hip Hop* as an expression of young black urban culture in the late 1980s. While both *Punk* and *Hip Hop* refer to identifiable popular music forms, they also refer to a much greater set of socio-political elements such as fashion, politics, race and age. Cavicchi (1998) presents an ethno-musicological study of how fans of Bruce Springsteen use his songs to shape identity and create community. Cavicchi suggests that the fan has a deeply uneasy relationship with the music industry. Whereas fans of Springsteen's music have a deep sense of personal commitment to the artist and his music, they are dependent on music labels for musical releases, artist information and access to the artist. They view the music industry as an obstacle rather than an enabler or a service provider. Cavicchi shows that the fans he studied disregarded the music industry, and created their own community-based information network using fan club newsletters, tape swapping and the Internet.

This aspect of cutting the music industry out of the loop has resonance with our earlier discussion on file-sharing networks. It is clear from Cavicchi's interviews that Springsteen fans do not view music as a commodity or a service, but as an integral part of how they give meaning to their lives. We will suggest that on-line music systems will need to cater for the fact that music is part of the fabric of social discourse, and that instead of looking to promote industry-to-consumer

relationships as they are currently doing, an alternative would be to enable listener-to-listener relationships.

2.5.5 Categories of Human Mark-up

We have identified three categories of human-based mark-up.

1. Inexpensive Meta Data:

This refers to content descriptions which can be obtained inexpensively. Generally catalogue metadata is freely available through services such as the CD database (www.cddb.com), or from the provider of the music service. CD Metadata typically includes data such as *artist name*, *label*, *release date*, *genre*, *contributing musicians*.

2. Knowledge Intensive Human Mark-up:

This refers to any mark-up in which people provide a detailed analysis of the music artefact so that it can be retrieved from a content- or knowledge-based retrieval system. This can be based on a formal analysis or an informal analysis. An example of formal analysis would be the musicological mark-up applied to music using the *Music Genome Project*TM of SavageBeast Technologies which we describe shortly.

The AMG's music content database³⁴ provides a mixture of formal and informal data:

MetaData: title, credits, release date, label, studio

Editorial Data: artist and album reviews, artist biographies

Descriptive Content: styles, keywords, tones, themes

Relational Data: similar artists, similar albums, roots and influences, other artists with whom the artist has worked.

However such knowledge does not come cheaply. A single user license is quoted at several thousand dollars per month (AllMusic 2002). Generally this type of mark-up is labour and knowledge intensive. AMG, for instance, employ an editorial team of 900 writers to keep on top of the popular music scene.

Distributed content gathering

Several companies have addressed the issue of music mark-up by encouraging on-line users to submit their own descriptions of music. For instance, the AMG Company also allows visitors to their website to provide feedback on how they view the style of music of a particular artist or band. The descriptors offered are not musical but impressionistic describing moods, emotion and sensations (see Figure 2.10).

³⁴ <http://www.allmediaguide.com/data.html>

Music Expert Check. If you know this artist well, your help in answering the following questions is much appreciated and will assist the AMG staff in improving the database. Do you feel this artist is:

	Some of Both		
Energizing, Exciting <input type="radio"/>	<input type="radio"/>	<input type="radio"/> Soothing, Relaxing	<input checked="" type="radio"/> N/A
Dense, Thick <input type="radio"/>	<input type="radio"/>	<input type="radio"/> Light, Free, Transparent	<input checked="" type="radio"/> N/A
Harsh, Aggressive <input type="radio"/>	<input type="radio"/>	<input type="radio"/> Gentle, Peaceful	<input checked="" type="radio"/> N/A
Cold, Firm <input type="radio"/>	<input type="radio"/>	<input type="radio"/> Warm, Soft	<input checked="" type="radio"/> N/A
Bright, Dynamic, Ormate <input type="radio"/>	<input type="radio"/>	<input type="radio"/> Low Key, Calm, Melancholic	<input checked="" type="radio"/> N/A
Popular, Plain, Simple <input type="radio"/>	<input type="radio"/>	<input type="radio"/> Elaborate, Sophisticated	<input checked="" type="radio"/> N/A
Dark, Pessimistic, Bitter <input type="radio"/>	<input type="radio"/>	<input type="radio"/> Light, Cheerful, Sweet	<input checked="" type="radio"/> N/A
Emotional, Sensual, Playful <input type="radio"/>	<input type="radio"/>	<input type="radio"/> Sober, Arranged, Proper	<input checked="" type="radio"/> N/A

Your name(optional):

Figure 2.10: AMG allows visitors to contribute content description to their site

Moodlogic provides a content service for *clients* who wish to organise their music collection. The Moodlogic client encourages users to mark up their music collections by allocating them activation points. While a song is played the user is asked a series of questions about the song's genre, instruments, lyrics, energy level, and mood. This information is distributed to other users of the Moodlogic service. However, the mark-up process in Moodlogic is non-trivial, requiring the user to select from several screens of quite abstract description (see Figure 2.11).

Please rate this song on each of the following scales:

Low Energy	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	High Energy
Constant Energy Level	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Changing Energy Level
Light Beat	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Heavy Beat
Slow Tempo	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Fast Tempo
Not Danceable	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Very Danceable

Figure 2.11: An excerpt from one of the screens offered to users by the Moodlogic client application

3. Usage data:

However, the most common and the easiest means of collecting music ‘mark-up’ in a distributed environment is by collecting user ratings. In this scenario users are simply required to rate a music item according to a numeric scale and these scores are then collected by the service provider (see Table 2.2). Ratings can be submitted explicitly or can be derived implicitly from the user’s on-line behaviour. This type of usage content is generally referred to as a ‘content-less’, in contrast to mark-up where clear semantics are attached.

Table 2.2: The typical format of usage data

	User A	User B	User C	User D	User E
Song 1	0.8	0.2	-	1.0	0.8
Song 2	0.2	0.8	0.6	0.4	-
Song 3	0.6	0.4	0.2	-	0.8

2.5.6 'Content-less' Retrieval

Automated Collaborative Filtering (ACF), a technique we discuss in detail in Chapter 4, is considered to be a 'content-less' form of personalisation (Balabanovic & Shoham 1997). Despite this label, music items *are* marked up in terms of the ratings allocated by users (see Table 2.1). The distinction is made because there are no formal semantics associated with the ratings given for a particular music item. We would suggest that this distinction is not entirely accurate, and that rating data, although noisy, implicitly contains meaning with respect to how the music item is valued among a population of users.

For example, although rating data does not enable an explicit query formulation, it does facilitate user A to query the system for songs that have been highly rated by users B, C and D. The selection of users suitable for User's A query is generally automated so the user's query simply becomes a request for the *n* songs most suited to his taste.

The advantages of 'content-less' recommendation in music are considerable. The knowledge elicitation and maintenance problems of content-based retrieval are avoided. (There is, of course, the 'cold-start' bottleneck which we describe in Chapter 4). The subtle distinctions that people make when making aesthetic judgments are very difficult to encode as formal knowledge into a content-based system. 'Content-less' recommendation implicitly captures such judgements in the ratings set it elicits from each user.

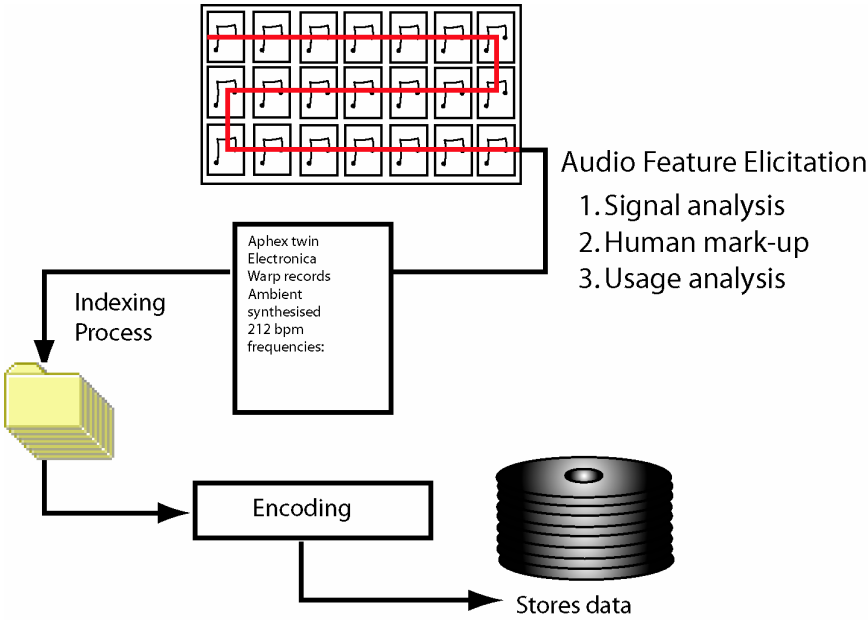


Figure 2.12: Three techniques typically used to procure description for retrieval of audio artefacts

2.5.7 Knowledge Expense

We have reviewed knowledge elicitation processes for audio such as automated signal analysis and techniques which require human intervention. These techniques require differing degrees of

knowledge engineering (see Table 2.3). Automated signal analysis can be classed as an expensive knowledge elicitation process. As well as being computationally intensive, this technique produces low level features which require human-guided feature selection techniques or human psychoacoustic models in order to produce features for retrieval/personalisation purposes. Furthermore, the extraction of audio features is still a new topic of research with no standard methodology in place. Techniques which require people to provide content description vary in terms of knowledge elicitation expense. Knowledge-based techniques are expensive, requiring the expert advice of a team of editors. For instance AGM employ over 900 freelance and staff writers to keep their database up-to-date.³⁵ On the other hand, acquiring Meta Data and Usage data are *inexpensive* techniques since the data can be obtained relatively easily.

Table 2.3: The table illustrates the knowledge expense for each annotation method

Knowledge Source	Knowledge Source	Knowledge Expense
Automated Signal Analysis	Machine and human analysis	High
Knowledge Intensive Mark-up	Expert human analysis	High
Meta Data	Human analysis	Low
Usage Mark-up	Human analysis	Low

2.5.8 Review of Music Personalisation Systems

Table B-1 in Appendix B provides a summary of the main players in the music personalisation services market. This appears to be a volatile market with several companies such as *Mubu* (mubu.com) and *Gigabeat* (gigabeat.com) going out of business in the past 2 years and another, Moodlogic, that has switched its service model. As we have shown in section 2.3.2, there have been few Digital Service Providers in the market until recently, and there is still much uncertainty about a working business model. This is an obvious factor for the difficulties for commercial growth in this area.

The first two companies we review employ the concept of music ‘DNA’ to suggest the unravelling of a unique set of attributes for music artefacts. This is a popular metaphor in this domain and is used by the CantaMetrix Company for its ‘MusicDNA’ technology, which is used to recognise pirated music, and by RealNetworks for its ‘Helix’ music delivery platform.

SavageBeast Technologies

SavageBeast technologies offer a content-based recommendation system which is powered by their proprietary *Music Genome Project*TM. This project involves building a database by manually marking up music tracks using expert human knowledge. Each track is analysed along 400 attributes – some of which are musicological (lyrics, instrumentation, compositional qualities) and others which are popular, informal attributes. SavageBeast state that they *do not* use automated

³⁵ <http://www.allmediaguide.com/data.html>

feature extraction techniques. The process is labour intensive as the company claims that each song is marked up by at least one of their music analysts.

Music Genome

The music recommendations service offered by Music Genome is based on research carried out in Stanford University and the Hebrew University in Jerusalem. Music is analysed using a “proprietary set of objective measurements” and assigned a “music DNA marker”. The Music Genome is able to build a profile of users by having them rate some music tracks. The profiling is based on “research in artificial intelligence and music cognition”. Although the company profile does not describe their technology, the last published research of the founders involved modelling the cognitive processes of a performing musician using symbolic and sub-symbolic logic (Gang et al. 1999). However, Music Genome does not appear to be active. The last press reports we could find (February 2001) suggested that Music Genome Software was being tested by Tower Records in Israel. We include the company here as the only example of a personalisation solution that claims to use some elements of music cognition.

MediaUnbound

MediaUnbound is based in Cambridge, Massachusetts and counts as an advisor, Patti Maes, one of the founders of the Firefly network and the Ringo music recommendation system (which is discussed in Chapter 4). MediaUnbound uses ‘content-less’ and content-based recommendation strategies. Their content representation involves manually building maps that correlate how close to each other various artists, sounds and songs are. Their core services are a song characterization system, a personalisation system, a selection engine and a playlist arrangement engine. MediaUnbound’s personalisation session is novel in that asks users to answer questions as well as provide ratings.

- If you were talking with a friend how you would describe your music preferences?
- What is your opinion of FM radio?
- What styles of music do you like?
- How familiar do you want your music?
- What phrases best describe your chosen style

Users are then offered a selection of individual songs and groups of songs to rate.

Although each question is posed separately, the questioning order is the same for each user so the personalisation strategy cannot be really classed as belonging to the *conversational* systems we review in the next chapter (Aha et al. 2001). MediaUnbound also provides a playlist arrangement engine which can arrange songs into playlists using heuristics proposed by their content editors. In May 2002 the company signed a deal to provide recommendation services on the PressPlay music service.

AgentArts

AgentArts sidestep the content issue by providing a general API for music recommendation and music management using content-less techniques and user profiling. The type of content description used is dependent on the implementer of the API. The demonstrations on the AgentArts site use simple music metadata and user demographic information. The AgentArts engine allows user preferences to be clustered into ‘styles’ which are described extensionally (see Figure 2.13). A similar technique is described in Clerkin, Hayes & Cunningham (2001).



Figure 2.13: User preferences represented extensionally in the AgentArts software

In the next section we introduce the basic functionality of the Smart Radio system. In Chapter 5 and Chapter 6 we will describe the personalisation techniques used in Smart Radio.

2.6 Introduction to Smart Radio

In this section we introduce the main features of the Smart Radio system, a system for personalised music delivery that has been in operation in the Computer Science Department, Trinity College Dublin since 1999. We will not go into great detail in this section as most of these aspects are discussed in more detail later in the thesis.

Smart Radio, although a prototype system, was the first personalised music service that demonstrated streaming playlists of music as a viable means of distribution (Hayes 1999, Hayes & Cunningham 2000). Our design goal was to provide a personalised service of streaming music using a recommendation system to suggest suitable compilations of music to listeners. The Smart Radio approach is to have people manage their music resources by putting together personalised music playlists. These playlists can then be recommended to other listeners using a combination of content-less and content-based recommendation strategies.

2.6.1 Streaming Service

The advantage of streaming a playlist is that it allows a programme of music to be delivered immediately. Putting together a compilation of music requires some effort and knowledge on behalf of the user. By making recommendations of playlists rather than music tracks we distribute this work to other members of the community.

Smart Radio recommendations are not anonymous. An early design goal of Smart Radio was to encourage social processes rather than replace them. We can see from the screenshot in Figure 2.14 that users can view the creator of each recommended playlist. Users can also choose to receive new playlists from other users that they nominate as trusted sources. The Smart Radio architecture is designed to allow users to build playlists as easily as possible, and automatically receive playlists from like-minded listeners in a seamless, transparent fashion. This facility encourages community participation by allowing users to know who their most consistent neighbours are.



Figure 2.14: A screenshot of a Smart Radio home page

2.6.2 Features

Recommendations

Smart Radio users receive recommendations from listeners who share aspects of their taste in music. This is the basis of automated collaborative filtering (ACF), a ‘content-less strategy’, which we will review in detail in Chapter 4. The recommendation strategy used in Smart Radio also uses a content-based technique to make recommendations that are sensitive to the user’s short-term listening interests. The content-based technique we use is case-based reasoning (CBR) which we review in Chapter 3. Recommendations are presented to the user as soon as they log in to the system. By default, the top recommendation is presented in the right most display panel. Users can scan the contents of each playlist quickly by clicking on the name of the list.

Playlist Organiser Panel

The main Smart Radio panel is shown in the left-hand side of Figure 2.14. It gives the user access to the following features:

Playlist Search: users can search the playlist database by artist, genre or track name. Each search can have up to three search criteria. For example, a user could search for a playlist that contains the genres, ‘folk’, ‘jazz’ and ‘blues’. The system will return the playlists in the system that best meet these criteria.

Your Playlists: This facility allows a user to view and retrieve the playlists he/she has played in the past.

Top Ten: The top ten playlists are calculated using a *sliding window* counter over the previous 7 days. The count is a reflection of what playlists are being used by users *other than* their creators. For example, a user who plays his/her own playlist repeatedly will have no effect on the top ten.

Neighbours: This feature allows the user to select users whose new playlists he/she wishes to view. As we will discuss later in this thesis, the ACF recommendation strategy will not recommend items that have not been rated by other users. By selecting ‘trusted’ neighbours, the user bypasses this problem. He/she will always have access to new playlists created by these neighbours.





New Playlists: The feature allows users to view the new playlists created by their selected neighbours within selected time intervals.

Preferences: Users can specify the preferred proportion of new items they would like to receive in each playlist.



Figure 2.15: Smart Radio playlist options

Playlist Options

As Figure 2.15 illustrates, the user has a number of options when he/she displays a Smart Radio playlist. Firstly, he can choose to play it, . By choosing the edit icon  a user can adapt the playlist to his/her taste. Or they can instruct the system to find more playlists similar to this playlist (). Finally, if the playlist is a recommendation, they can reject it by choosing the  icon. The user can also get an explanation as to *why* the playlist was recommended by rolling over the *(why?)* text.

Player controls

The player controls are permanently located at the top of the Smart Radio page. The panel on the right will indicate the status of the player (opening a playlist, buffering a song file, or as shown in Figure 2.16, the title and artist of the song current playing). The control panel on the left allows the

user to stop, pause or fast forward/rewind any current song. The user can also quickly adjust the overall volume from the system.



Figure 2.16: Smart Radio controls

2.6.3 Editing a Playlist


The *Edit* mode  of Smart Radio allows the user to modify a previously existing playlist or build a new playlist from scratch. When editing a playlist, users can remove, replace and change the position of tracks.



Figure 2.17: A playlist in *Edit* mode

Adding/replacing tracks



The *artist-replace* icon  and the *genre-replace* icon  allow users to quickly replace individual tracks. By Clicking on the *artist-replace* icon, the music track to the left of the icon can be replaced with another track by the same artist. By clicking on the *genre-replace* icon, the music track on the left of the icon can be replaced with another track by the same genre. A single click on one of these icons will remove the track to the left and, at the same time, bring up the **search pop up window** from which the user can choose from a selection of replacement tracks either by the same artist, or within the same genre. For example, clicking on the *artist-replace* icon beside the track ‘This Years’s Love – David Gray’ in Figure 2.17 will create the two windows shown in

Figure 2.18 below. The slot left by the removed track can be filled by clicking on one of the tracks offered in the search window.

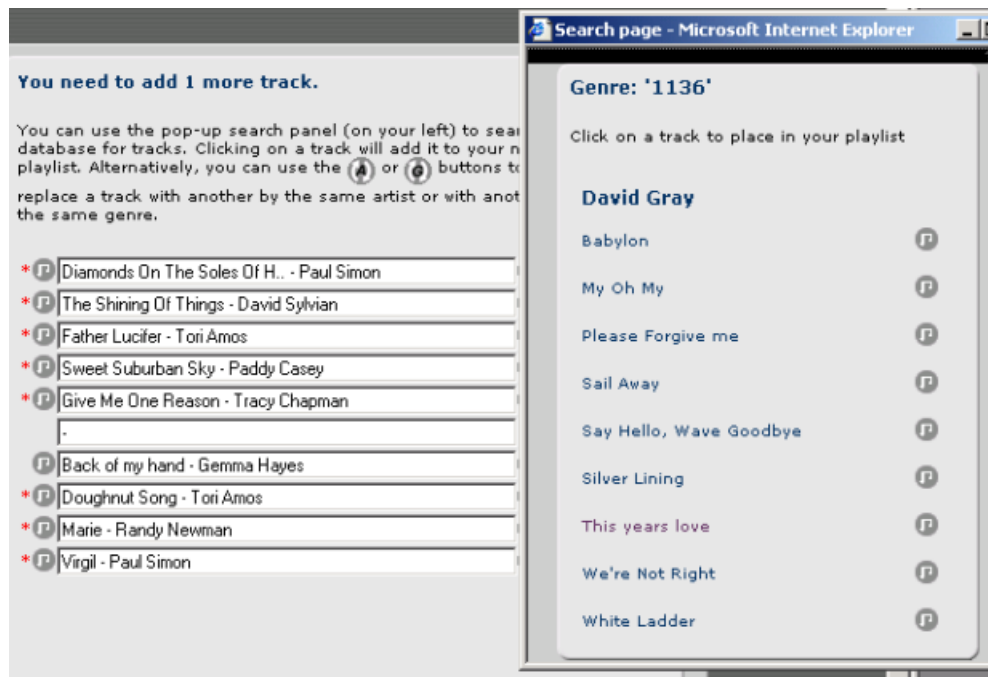




Figure 2.18: Choosing the *artist-replace* icon to remove one David Gray song and replace it with another

Additionally, users can find music tracks to add to a playlist using the 'Search Tracks' button  shown in the right of the window in Figure 2.17. Clicking on this button will bring up a search window that will allow users to search for tracks by *genre*, *artist* or *track name*. As before, clicking on the track will add it to the current playlist.

Naming the Playlist

Once the playlist contains ten tracks the user is prompted to name it and set it to play. By default, the system assigns a playlist name consisting of the user name and the date and time. Many listeners choose to nominate much more colourful titles.

Audio Samples

When compiling or editing a playlist a user can choose to hear a 30 second sample of any song by clicking on the music sample icon .

2.6.4 Rating Items or Artists

Users can rate music tracks on a scale of 1 to 5 by clicking on one of the 'smiley' icons shown next to song titles in Figure 2.14. Users can rate an artist *at any time* by clicking on the artist name when it appears in a playlist. This activates the artist-rating pop-up window (see Figure 2.19).

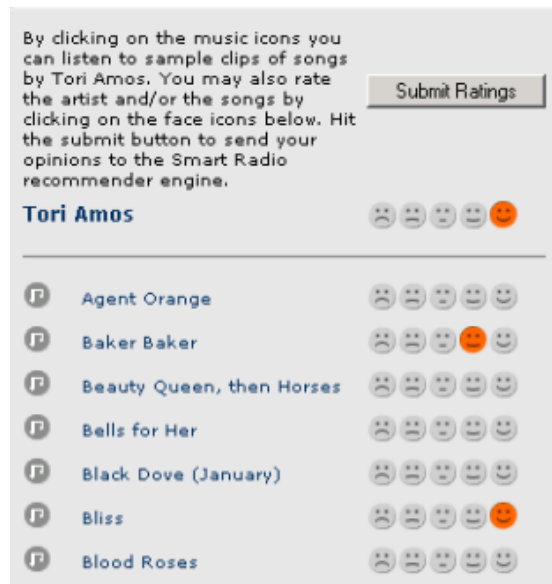


Figure 2.19: The artist rating pop-up box

2.6.5 Smart Radio Music Library

Smart Radio contains 4131 tracks from 333 artists. The tracks were ripped into mp3 format from the personal CD collections of colleagues and friends. The tracks are marked up with simple meta-data tags (*artistName*, *album*, *genre*) extracted during the ripping process. This data was taken from the CD database (www.cddb.com). There are 23 genres represented in the database, not in equal proportions. Given that the information in the CD Database is provided voluntarily, many of the *genre* tags were inaccurate. By inspection we re-labelled about one quarter of the songs. We also created a new ‘songwriter’ genre which characterises solo songwriters who did not fit comfortably into rock or the folk genre. Indeed, genre is a very rough way of classifying music. Many artists do not fit well within the genre assigned to them and we could have split many genres into sub-genres to capture the nuances of an artist’s style. However, we decided that this process would ultimately be unending, and to work with ‘light’ representations we could glean from the music files.

2.6.6 Copyright

In Ireland, the Irish Music Rights Organisation (IMRO) administers the performing rights copyright for music on behalf of its members. Music users such as broadcasters, venues and businesses must pay for their use of copyright music by way of a blanket licence fee. Performing rights agencies, such as IMRO, operate in given geographical regions to collect these fees and distribute them to the copyright owners involved. The monies earned by copyright owners in this way are known as public performance royalties. Performing copyright is separate to mechanical copyright and recording copyright which we have mentioned earlier in the chapter.

In 1999, we approached IMRO about obtaining a licence to operate an ‘experimental Internet-based radio system’. However, they had no licensing strategy in place for web casting.

Instead they suggested that we ‘broadcast’ only within the confines of the department, and as such, they would waive the licensing issue. However, within 12 months the Napster network was forcing copyright agencies to take a hard stance on digital media. Had we approached them after this point, it is probable we would not have gained permission, although informal, to operate.

In fact the whole issue of web casting is a vexed one. We were quite naive in believing that a single performing rights licence fee would allow us licence to operate. In fact we should have approached each record label represented in our music database in order to license the individual music tracks. Since the labels at this stage had not decided on digital licensing or distribution models, it is unlikely we would have succeeded. Furthermore, we should have also approached the national mechanical copyright organisation for a license. In order to generate smooth playback of incoming streams, a computer temporarily buffers some of the data in memory. Music publishers have stated that the data in this buffer should be considered a physical creation that would require web-casters to pay a mechanical royalty, similar to what they pay for downloads or CDs.

The research described in this thesis has only been possible because Smart Radio has operated ‘under the radar’ for the past few years. The stringency which is being applied to music licensing today is a reflection of the issues at stake for all sectors of the music industry. However, it is unlikely that research projects like Smart Radio could be initiated today with these licensing restrictions.

DRM

When we started this project, digital rights management for audio was nonexistent. We chose to use streaming media because it allowed us to pursue the metaphor of personalised radio but it also allowed us to avoid the problem of users downloading all our music files at once and never returning. Our streamed audio is not encrypted, so it is not piracy proof, although DRM protected streams are regularly compromised. A stream can be saved to disk with the right tools and a little know-how. However, we feel that it is easier to download a file from a file-sharing network than it is to hack a Smart Radio stream.

2.7 Conclusion

In this chapter we have reviewed the issues involved in the on-line distribution of music. We see that the file-sharing networks have created a crisis for the music industry by setting up a distributed distribution network that does not pay the industry for its ownership of copyright. The industry faces another obstacle in the sceptical attitudes of its consumer base who feel that they have been overcharged for years. We argue that the file-sharing networks are attractive not just because they offer the largest selection of music available anywhere, and for free, but *because* they are outside the influence of the media conglomerates who dominate the industry. In the US, the RIAA representing the music industry is doing irreparable harm to the credibility of the industry by

pursuing individuals for breach of copyright. Adding to this is a recent court settlement in the US which found the major record companies guilty of price fixing.

We suggest that rather than attempting to position themselves as on-line retailers with zero credibility, they re-examine what has made file-sharing a popular enterprise. The architecture of Smart Radio offers a compromise arrangement. It enables users to swap playlists of music with each other in a transparent manner. From the service provider's point of view, such an arrangement would fit neatly into the architecture of a system such as Listen.com's Rhapsody system. Thus media owners could take on the role of content provider, licensing content to independent music services/retailer.

Playlists are swapped in Smart Radio using personalisation techniques, where the system learns the preferences of the user. We have seen that there are two types of personalisation techniques – 'content-less' and content-based. We suggest that the distinction between the two is not so apparent. Content-based techniques for music are difficult to build because of the knowledge elicitation bottleneck. Mark-up must be automatically extracted using signal analysis or based on human analysis. We see that there are roughly two types of human analysis – one which uses formal metrics and another which use more subjective, informal metrics. 'Content-less' techniques belong to the latter category, making use of subjective user ratings to mark-up music. Although less immediately interpretable than content-based mark-up, rating data has an advantage in that it is generally easier to acquire than content-based mark-up.

In the next chapter we examine Case-Based Reasoning as an example of a 'content-based strategy' for making personalised recommendations. In Chapter 4 we will contrast this with Automated Collaborative Filtering, a 'content-less' strategy.

Chapter 3: Case-Based Reasoning

Case-based reasoning (CBR) is a problem-solving paradigm that is in many ways different from other AI approaches. Traditional AI approaches rely on general knowledge of a problem domain, and tend to solve problems on a first-principles, or ‘from scratch’ basis. CBR systems solve new problems by utilising specific knowledge of past experiences, and this knowledge is encoded within a corpus of previous problem-solving episodes called a case base. Another important difference is that CBR supports incremental, sustained learning, since a new experience is retained each time a problem has been solved, making it available for future problems.

CBR has proven itself to be a methodology suited to solving *weak theory* problems, that is, problems where it is difficult or impossible to elicit first principle rules from which solutions may be created. In such domains, it is an attractive reasoning alternative to rule-based systems because of the potentially lower knowledge engineering costs (Cunningham 1998). As a result, it is a reasoning methodology which has been successfully used in a variety of academic and industrial applications. Areas to which it has been applied include

- Classification
- Diagnostic tasks
- Configuration and Design
- Planning
- Decision Support
- Information Retrieval
- Personalisation

CBR has found increasing application on the Internet as an assistant in e-commerce stores, as a reasoning agent for product configuration and as a strategy for user personalisation. An example of the latter is case-based user profiling in which the interests of the on-line user are captured and used to make recommendations based on the similarity of the profile to items such as TV programmes (Smyth & Cotter 1999 a,b), job descriptions (Bradley et al. 2000) or flight offers (Coyle et al. 2000). In this chapter we review CBR, with an emphasis on how it is used in on-line applications.

3.1 Roots of CBR: AI and Cognitive Science

From the beginning, AI research (Turing 1950, Minsky 1961) has attempted to replicate human problem-solving competence in machines. While such reasoning is often associated with chaining generalised rules together, CBR involves reasoning from a memory of stored cases which record prior specific episodes.

A case-based reasoner solves new problems by adapting solutions that were used to solve old problems (Riesbeck & Schank 1989)

CBR studies have their roots in research in cognitive science. The work of Roger Schank is often held to be a significant starting point (Schank & Abelson 1977, Schank 1982). Schank hypothesised that our general knowledge about situations is recorded in the brain as scripts, which allow us to set up expectations and perform inferences. According to this theory, a person when faced with a new situation firstly attempts *situation assessment* in which the key features of the new situation are extracted. The retrieval of the appropriate script is then attempted. The script needs to be indexed in a way that it can be accessed using the features describing the new situation. Finally, the script may be adapted to cater for the specifics of the new situation. However, later findings showed that this is not a complete theory of memory representation (a person may mix scenes from similar situations, e.g. going to a doctor's office and going to a dentist's office). The key point is that CBR is based on the *reminding* of similar prior episode/s to solve a current problem or classify a situation.

3.2 CBR Knowledge

This problem-solving strategy is dependent on two tenets. The first is that the world is a *regular* place in which similar problems have similar solutions. The second is that the world is a *repetitive* place in which similar problems tend to recur. Instead of attempting to model the causal interactions that define a particular domain, the idea in CBR is to retrieve and adapt cases when solving new problems. We illustrate this in Figure 3.1 where SP represents a specification of a problem, SL is a solution to that problem and FP is some hypothetical First Principles reasoning that would infer the appropriate solution for the problem description SP. The idea in CBR is to avoid having to model this First Principles reasoning by retrieving a case with a similar description SP' instead and adapting the solution to that case (SL') to fit the problem at hand. The implication is that this retrieval and adaptation process is easier to implement than deriving First Principles by which to solve the problem.

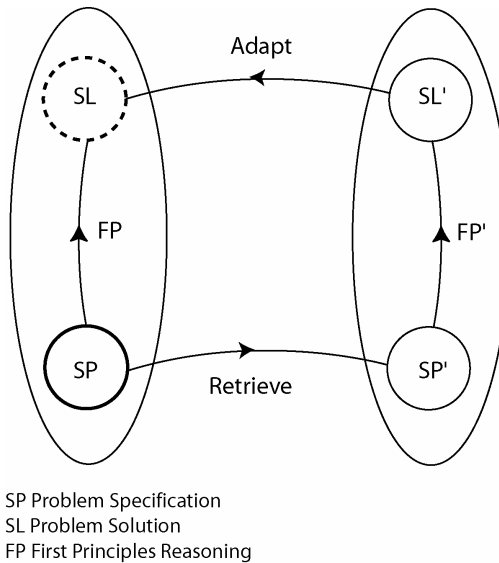


Figure 3.1: The transformation mappings used in CBR

Thus CBR is often credited with providing an alternative solution to the intensive knowledge engineering requirements of rule-based or model-based systems. However, for this to hold true the knowledge engineering involved in building a case base must be low. In many situations case data can easily be obtained. This might be in the form of problem-solving episodes recorded in diagnostic logs, or in the form of catalogue or configuration data. In such situations a CBR system can be much more quickly deployed than a rule-based system. However, where the development of a CBR system also involves a deep analysis of the domain in order to produce a case base, the advantage of CBR over knowledge intensive strategies is not so obvious. Despite this, it could be argued that in many situations only a CBR approach is viable. This occurs in ‘weak theory’ domains where it is difficult or impossible to elicit first principle rules from which solutions may be created. In such domains human experts would typically reason from precedents rather than from first principle.

CBR is a *lazy learning* technique (Aha 1997b), deferring reasoning until query time. This makes it suited to domains where the case base is being added to or edited frequently and where it would be difficult to create a model that accurately reflected the state of the changing environment.

3.2.1 Knowledge Containers

In rule-based systems knowledge is encoded in the causal relations specified by each rule. Richter (1995, 1998) defined four ‘containers’ which hold the knowledge used in a CBR system.

1. **Vocabulary knowledge** consists of the semantic components describing the domain that can be manipulated by a reasoning system.
2. **Case Knowledge** consists of the ‘problem’ episodes or instances represented as cases that can be used to solve similar problems in the future.
3. **Similarity Knowledge** represents the similarity measures which are used to match cases in a particular domain.

4. **Adaptation Knowledge** is knowledge used to adapt the solution of the matching case for the target problem.

Richter suggests that CBR's advantage is due to the fact that while *vocabulary*, *similarity* and *solution* knowledge must be structured and encoded during compilation time, the knowledge contained in the cases is not generalised to form rules, but instead lazily invoked at query time.

Figure 3.2 illustrates the knowledge containers involved in the earlier example. We also show in this example that CBR can be viewed as exploiting the relationship between two different types of similarity: the similarity used in the problem space and the similarity used in the solution space. Whereas similarity knowledge in the problem space is relatively easy to engineer, 'similarity' in the solution space may be based on solution knowledge which may be difficult to model. We will discuss this issue more concretely when we discuss the CBR *reuse* phase.

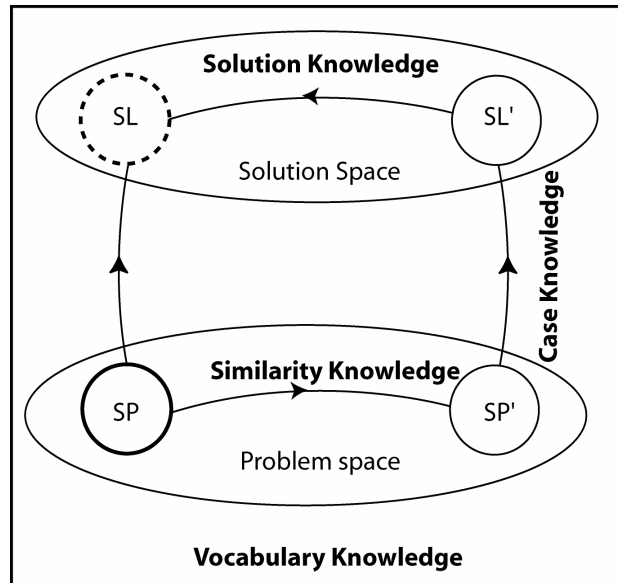


Figure 3.2: The knowledge containers used in the CBR process

3.3 Methodology

The CBR methodology is often expressed in terms of a runtime cycle that contains four key phases (Aamodt & Plaza 1994). These are known as the *four Rs* (see Figure 3.3).

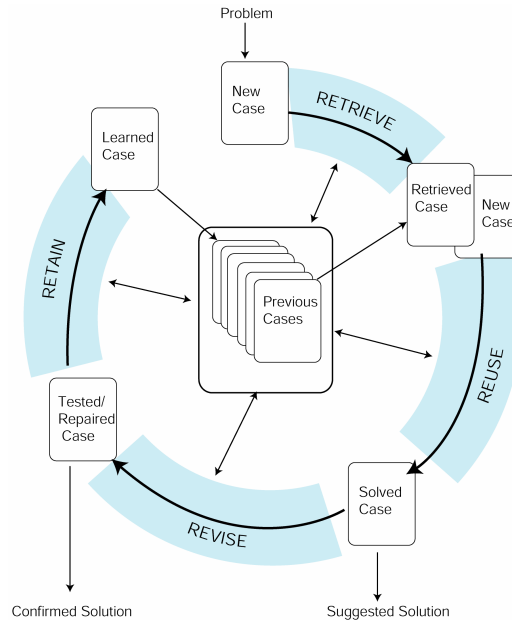


Figure 3.3: The CBR-Cycle from Aamodt and Plaza (1994)

Retrieve the most similar cases.

Reuse: solve the new problem using the knowledge and information in the retrieved cases to solve the new problem.

Revise: evaluate the suggested solution.

Retain the parts of this experience which can be used to solve new problems in the future.

The *four Rs* describe the naive view of CBR at runtime. This model is useful in that it partitions the issues involved in designing a CBR system. However, as we shall discuss in the sections below, each phase receives different degrees of attention in applied CBR. The *retrieval* phase, for instance, is generally considered to be the most important as problem solving relies upon finding similar case descriptions in the case base. The *reuse* phase is problematic in that it relies upon first principles in order to adapt solutions. The *revise* phase receives little attention and is often considered an extension of the *reuse* phase. The *retain* phase is often not engaged at runtime. This is the learning phase of CBR and many of the techniques employed are carried out off-line. We will discuss each phase in more detail in the following sections.

3.4 Case Representation, Similarity and Retrieval

3.4.1 Case Representation

A case is a representation of a particular situation or instance. Unlike a generalised rule, a case records several features and the specific values that occurred in that situation, and is independent of other cases in the case memory. As we have suggested, it typically consists of two components: a specification part and a solution part. The specification part consists of a set of attribute value pairs that should uniquely define that case and be sufficient to predict the solution for that case. In many situations, a flat attribute value list is adequate for case representation. Table 3.1 illustrates a simple flat case from the travel case base (Lenz 1993). The solution in this context is the *Hotel* field. However, a case does not always need to specify a solution. For instance, a case describing a product configuration may be a solution in itself.

Table 3.1: An example of a flat attribute-value case representation from the travel case base

Case	Journey149
HolidayType:	Recreation
Price:	922
NumberOfPersons:	3
Region:	BlackForest
Transportation:	Car
Duration:	7
Season:	August
Accommodation:	ThreeStars
Hotel:	Berghotel Kandel, Black Forest

Representations other than the flat feature vector are possible. Lenz et al. (1998) use a free text representation to represent text-based documents. The *NaCoDAE* system of Aha et al. (2001) uses a list of questions and answers for each case instance. Object-oriented representations are generally used for complex or structured case knowledge. In such a representation, a case consists of a set of attribute objects which in turn have their own set of features objects. The object-oriented structure of the CASUEL case representation language is an example (Manago & Bergmann. 1994). While such representations allow taxonomic relations and inheritance features to be modelled, similarity computation and retrieval are accordingly more complex (Bergmann & Stahl 1998). Gebhardt (1997) provides a review of such structured case representations. Sanders et al. (1997) argue that a case should be represented as a graph structure, that is, as a set of nodes and arcs which allow the relations between objects in a case to be fully modelled. We will concentrate on the feature vector representation in this chapter since it is the most commonly used and the most pertinent to our domain.

3.4.2 Similarity and Retrieval

Retrieval is a key issue in CBR research. It rests upon the idea of a *similarity measure* which is a function to compare two cases. Similarity is not an absolute notion but is always dependent on

what the similarity between two cases is used for. This is referred to as the *utility* function of the domain (Althoff & Richter 1999). Very often similarity is calculated as an inverse function of distance.

If we consider two feature vectors $X = (x_1, \dots, x_i, \dots, x_n)$ and $Y = (y_1, \dots, y_i, \dots, y_n)$. There are two aspects to similarity

- **Local similarity:** $\text{sim}_{A_i}(x_i, y_i)$ which refers to similarity at the feature level.
- **Global similarity:** which similarity refers to overall similarity at the case or object level. The global similarity function is an amalgamation function of the local similarity measures.

Equation 3.1

$$\text{sim}(X, Y) = F(\text{sim}_{A_1}(x_1, y_1) \dots \text{sim}_{A_n}(x_n, y_n)).$$

For real valued attributes this amalgamation function may be the same across all attributes, such as in the city block measure (see Equation 3.2)

Equation 3.2

$$d(x, y) = \sum_{i=1}^n |x_i - y_i|$$

or a weighted function such as weighted Euclidean distance (see Equation 3.3).

Equation 3.3

$$d(x, y) = \sqrt{\sum_{i=1}^n w_i \cdot (x_i - y_i)^2}$$

In both these cases the local similarity measure, $\text{sim}_{A_i}(x_i, y_i)$, is the absolute difference $|x_i - y_i|$ which is a symmetric measure.

However, local similarity functions may be more complex. For example, with real valued attributes similarity can be an asymmetric function of the difference between attribute values. Stahl (2002) illustrates this with a simple example in the domain of PC sales where each case represents a computer configuration (see Figure 3.4). In the example, Q is the query specification and C the target case, which each have three attributes: *Price*, *CPU-Clock* and *CD Drive*. The local similarity measure for the *price* attribute is an asymmetric ‘less is perfect’ function where the optimal similarity score is achieved for that attribute if the target price is less than the price specified in the query. Contrary to this, the local similarity score for the *CPU-Clock* is a ‘more is perfect’ function where the optimal score is that achieved if the target CPU is greater than that specified by the query. Up to this point we have talked about real valued attributes; the CD Drive is an example of attribute whose type is an unordered symbol. For such types, similarity is defined by hand and encoded into a similarity table (see Figure 3.4). On the other hand, the similarity relationship between ordered symbols (a taxonomy, for instance), can be encoded into a tree or graph structure

where similarity is a function of the distance between nodes (Bergmann 1998). We will give an example of this later in this chapter in relation to the CASPER system.

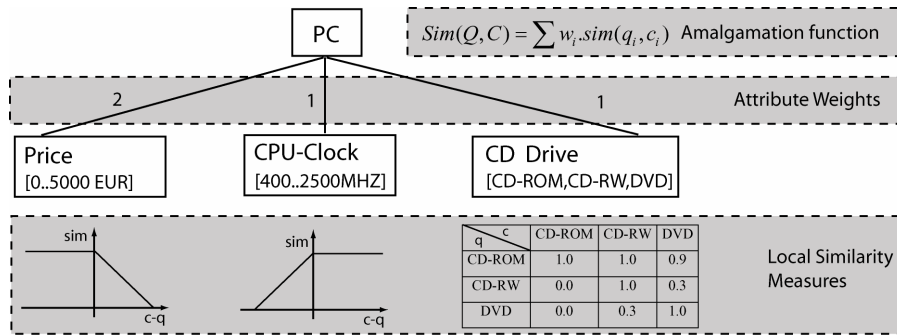


Figure 3.4: Defining similarity measures for the domain of PC sales (from Stahl 2002)

Weighting

The amalgamation function for this domain is a weighted average of the local similarity measures.

Equation 3.4

$$Sim(Q, C) = \sum w_i \cdot sim(q_i, c_i)$$

The weight attached to each attribute is an indication of its relevance. Since k -NN is very sensitive to the presence of irrelevant or noisy features, it is important to lower the weight for these features. Automatic learning of feature weights is a well established topic in machine learning and CBR (Wettschereck & Aha 1995). Introspective learning of feature weights, for instance, is a technique which monitors the problem-solving performance of the reasoning system in order to increase or decrease the weights of selected case features. This technique has been used to determine weights for each feature in each case in the case base (Bonzano et al. 1997). Feature weighting has recently been linked to the notion of *utility*, a term which refers to the context in which similarity is used (Bergman et al. 2001).

For instance, one aspect of utility is that it is dependent on the preferences of the user conducting the search. Thus feature weighting should reflect those preferences. This is particularly pertinent for e-commerce situations where the user may have specific ideas of which features are important. In the example above, price has the greatest weight. However, another on-line user may want the highest CPU-clock speed possible at all costs. Stahl (2001) describes how user-specific weights can be learned based on how the user perceives the correct ordering of a retrieved set of cases. In Chapter 5 we will describe how we set feature weights on the fly in Smart Radio based on the composition of the target playlist.

Retrieval

The Retrieval step is fundamental to CBR. While a database's search uses *equality* to find matching records, retrieval in CBR uses a search for the *nearest neighbour*. There are different approaches to

retrieval which depend on the case representation, the complexity of the similarity computation and the size of the case base. We can divide these approaches into three categories:

- k -Nearest Neighbour
- Index-based approaches
- Database approaches

k-Nearest neighbour

The k -Nearest Neighbour approach (k -NN) involves defining a similarity measure (such as shown in Equation 3.2 or Equation 3.3) so that two cases can be compared for similarity. Retrieval proceeds by sequentially comparing the query case to each case in the case base, and returning the k most similar cases. The solution of the target case can be constructed from the solution information in the k -nearest neighbours. If the solution is a class, for instance, majority voting could indicate the class of the target query. The key advantage of k -nearest neighbour is its simplicity. It does not require additional indexing structures to be created for retrieval and the case base can be incrementally extended without recompiling the memory structure. Furthermore, any metric can be used to measure similarity. The major drawback is its scalability. Retrieval time increases linearly with the size of the case base. Thus k -NN is unsuited to domains (such as an e-commerce store) where there may be thousands of cases and where query response time is important.

Index-based retrieval

Index-based retrieval involves pre-computing indexes which allow rapid access to the portion of the case base most similar to the target query. The most commonly used indexing technique in CBR is the k -D-tree (Bentley 1975, Wess et al. 1993) which is typically a binary tree in which the maximum depth is the number of attributes used in retrieval (k) (However, if an attribute is represented more than once the depth of the tree will be greater than k).

The key idea of this approach is to build a tree which splits the search space into a number of parts which contain similar cases according to the similarity metric. The root node of the tree contains all the cases in the case base. Each inner node of the tree represents a subset of the case base which is further portioned into disjoint subsets. Each node contains the bounding values for each attribute. Figure 3.5 gives an example of a k -D-tree for a case base in which each case has two attributes, A_1 and A_2 . This example is taken from Wess et al. (1993).

CaseBase={A,B,C,D,E,F,G,H,I}

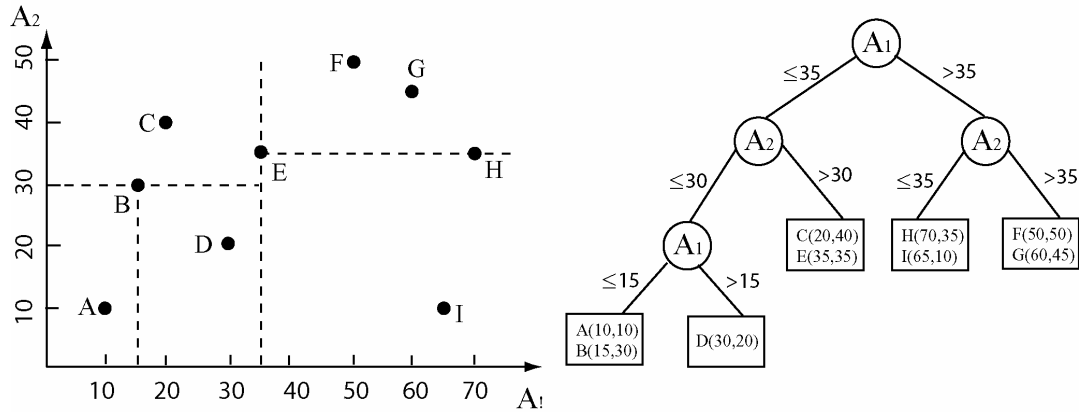


Figure 3.5: An example of a k -D-tree for CBR (from Wess et al. 1993)

The key issue in building a k -D-tree is choosing the indexes which will split the case base as efficiently as possible. The root node is assigned to the attribute which best partitions the case base into two equal parts. This process is continued recursively until a few cases are left which are stored together at the leaves. The determining of the partitioning attribute is crucial to improving retrieval efficiency with this approach. The partitioning of the search has to reflect the structure and density of the case base. Wess et al. (1993) use *inter-quartile distance* as a partitioning measure which is a statistical measure which can be used to reflect the distribution of the instances with respect to the similarity measure being used. Quartiles split a given distribution into four equally sized areas. The inter-quartile distance is taken as being the distance between the first and the third quartile. Thus, during tree building, for each partition the attribute with the greatest inter-quartile distance (based on the similarity measure) is selected as the most discriminating available.

Another partitioning measure that can be used is *Entropy* (Quinlan 1986, 1993), which is a measure of the impurity of an arbitrary collection of examples. It is given by

Equation 3.5

$$\text{Entropy}(S) \equiv \sum_{i=1}^c -p_i \log_2 p_i$$

where S is the set of instances, c is the number of possible classes and p_i is the proportion of S belonging to class i . If S has equal proportions of each class, entropy is 1. If S contains only one class, then entropy is 0. *Information gain* is the expected reduction in Entropy caused by partitioning the case base (S) according to a particular attribute. In decision tree algorithms such as ID3 (Quinlan 1986) or C4.5 (Quinlan 1993) attributes with high information gain are chosen to act as indexes higher up the tree. The disadvantages of using information gain are that the distribution is not measured with respect to the similarity function on which the retrieval is based, and it requires each case to have been assigned a class label which may not be appropriate for certain domains.

While Indexing does significantly speed up retrieval it does have some disadvantages. D-trees in general do not handle problems with missing values very well. There are also some limitations to the types of similarity function that can be used (Wess et al. 1993). The tree will have to be rebuilt from scratch when new cases are added. In Section 3.8.4 we examine indexing again for systems in interactive CBR in which missing values are a standard part of operation.

Database approaches

Another approach to retrieval is to build a CBR system on top of a relational database (Shimazu et al. 1993, Schumacher & Bergmann 2000). This technique uses a two-stage retrieval in which a wide net search is performed by querying the database, after which a similarity-based retrieval is carried out on the returned result set. This type of retrieval assumes that cases are evenly distributed across the n -dimensional attribute space, and that a weighted average is used for the global similarity metric. Figure 3.6 illustrates Schumacher & Bergmann’s (2000) implementation which uses a series of *query relaxation* stages (illustrated by the grey rectangles around the query point Q) to extract candidate cases from the databases. Query relaxation involves making a series of SQL queries to the databases and calculating the similarity of the returned cases. This is repeated until ‘sufficient cases’ have been retrieved.

The k most similar cases lie within a kind of hyper-rhombus in n dimensional space (illustrated in 2 dimensions by the dotted line in Figure 3.6) whose size is determined by the least similar of the k cases. The goal is to approximate the similarity-rhombus using the rectangular SQL-queries. The rectangular ‘rings’ around the query are increased until k cases are retrieved.

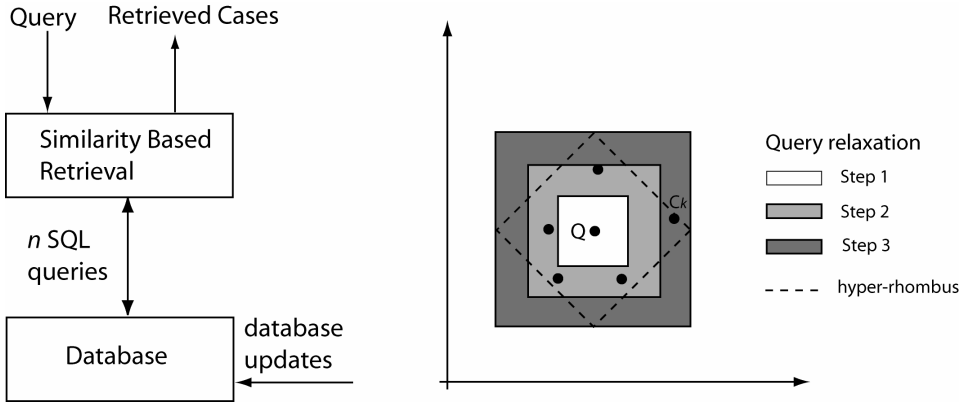


Figure 3.6: Query relaxation

The key issues for this type of retrieval are the SQL query formulation stage, which must initially retrieve all possible matches, and the choice of stopping criteria.

This technique has the advantage that the database representation does not have to be duplicated by the case base memory. Instead, candidate cases are retrieved from the database on a query basis and k -NN retrieval performed. Furthermore, this type of retrieval can be performed on any database. However, if the SQL query formulation is not handled correctly, sufficiently similar cases may not be retrieved from the database. If the query relaxation ‘ring’ is too large, too many

cases may be retrieved resulting in a slow second stage (k -NN) retrieval. On the other hand, if the ‘ring’ is too small it will result in too many consecutive SQL queries.

3.5 Reuse

Once a matching case (or an ordered set of matching cases) is retrieved, the next stage in the CBR cycle involves reusing the solution(s) of the retrieved case(s). In many CBR systems the solution to the target problem can be taken unchanged from the most similar retrieved case. If the problem is a classification problem, majority voting can be used to provide the class. In other instances the solution of the retrieved case must be *adapted* to the requirements of the target specification. As we illustrated earlier, CBR makes use of the relationship between the problem space and the solution space. Typically the adaptation stage employs a set of transformation rules for the solution (This is the *adaptation knowledge* described by Richter). These rules are employed to modify the retrieved solution based on the value difference in the attributes of the target case and the retrieved case. This type of adaptation is known as *Transformational Adaptation* (Wilke & Bergmann 1998). It can be characterised further into *substitutional* adaptation in which the values of the solution attributes may be changed, and *structural adaptation* in which elements of the solution may be reorganised, deleted or added to.

Derivational adaptation, on the other hand, involves replaying the derivation of the retrieved solution in the context of the target problem. The type of knowledge required for derivational adaptation is different from that described for transformational adaptation, which concerned modifying the solution based on differences between the two cases. Instead the solution for the retrieved case contains a trace of how the solution was constructed from scratch. This trace can be applied to the retrieved problem context and a new solution generated.

Compositional adaptation involves combining adapted components from multiple cases to produce a new composite solution.

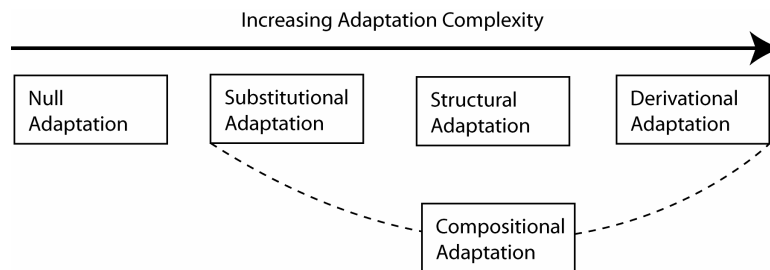


Figure 3.7: The adaptation continuum (based on Wilke, Smyth & Cunningham 1998)

Perspective on adaptation

It is generally agreed in CBR research that adaptation poses as many problems as it appears to solve. Whether adaptation is transformational or derivational, adaptation knowledge is based on rules which encode the causal links between problem and solution. This means that CBR developers have to confront the knowledge acquisition bottleneck which CBR was supposed to

circumvent in the first place. Of course there are different degrees of adaptation complexity (see Figure 3.7). The rules required to implement substitutional adaptation may be trivial compared to those required for derivational adaptation, in which first-principles domain knowledge is required to produce the solution trace. Furthermore, the adaptation phase may be a computationally complex phase, particularly in domains that require structural or derivational adaptation such as design or planning. In a worst case scenario, Nebel and Koehler (1995) show that plan reuse is actually more complex than generating plans from first principles. Smyth and Keane (1998) suggest assessing adaptation complexity during the retrieval stage of the *Déjà Vu* software design planner before having to incur the full cost of adaptation. Thus in the *Déjà Vu* system design plans which are more easily adaptable within the context of the design problem are preferred.

The problems associated with adaptation have led most developers of applied CBR systems to bypass this phase entirely (*null adaptation*). Instead such systems use the retrieval facility of CBR to propose cases for the human user to adapt and evaluate. Some systems take a middle ground approach suggesting aspects of the solution that the user might change or revise (Leake 1996). A similar approach is taken by the Smart Radio system which flags songs in a playlist that the user may wish to replace.

3.6 Revise

The Revise phase of CBR concerns the evaluation of the applicability of the proposed solution in the real world. The criteria used for this include:

- Solution correctness
- Solution quality
- User preferences

This phase has generally been considered fairly trivial in CBR research and very little work has been done on automatically verifying the proposed solution. Instead, the *Revise* and *Reuse* phases are often considered a single user adaptation phase.

However, in Section 3.8 we discuss interactive CBR systems in which the user iteratively provides feedback. We can consider such feedback to be a part of the *Revise* phase in which the user, by evaluating partial solution correctness, helps the retrieval system to produce an optimal solution. In some situations such as learning attribute weights for a particular user from case ordering feedback, the revise phase involves evaluating solution correctness (case ordering) in order to provide better retrieval results for that user in the future.

3.7 Retain

The Retain phase in CBR is concerned with preserving those parts of the problem-solving experience that are expected to be useful for future problem-solving sessions. This is the learning phase of CBR because its objective is to improve the problem-solving competence of the system. Trivially, we can improve this competence by adding each newly solved case to the case base. Thus

by increasing the number of cases in the case base we have a better chance of matching future problem cases without having to perform adaptation (or at least with minimal adaptation). This is an example of naive runtime learning suggested by Figure 3.3. Adding cases indiscriminately to the case base leads to a *utility problem* particular to CBR whereby retrieval efficiency begins to decrease once the case base has reached ‘saturation point’. After a certain point the gains to be made by not having to perform adaptation are lost as retrieval time increases (Smyth & Cunningham 1996). As CBR is deployed in business and industry, it has become important to examine how case bases can be updated and maintained. Thus research has been carried out on how to retain retrieval efficiency while preserving the problem-solving potential of the case base. These techniques are generally not performed during run-time, and can be considered a separate maintenance or learning phase associated with the Retain phase.

3.7.1 Off-Line Learning

Smyth and Keane (1995) proposed a competence measure for deleting cases from a case base. The *competence* of a case can be characterised by two properties: *coverage* and *reachability*. The coverage set of a case is the set of all target problems that the case can be used to solve. The reachability of a target is the set of all cases that can be used to provide a solution for the target problem. In order to characterise the case base in this way, the distribution of possible problems is assumed to be *a priori* given by the cases in the case base. Conversely, another related approach is to build the case base from scratch by selecting only training instances that contribute to performance, a process which is known in machine learning as *editing*. Smyth and McKenna (1999) adapt the *Condensed Nearest Neighbour* (CNN) editing approach for use in case-based systems. They define a new competence measure called *relative coverage* which takes into account the local coverage of the case as well as the degree to which this coverage is duplicated by neighbouring cases. This measure allows cases to be sorted according to their possible competence contributions. Using the CNN case selection approach, competence-rich cases are then selected before less competent cases, thereby maximising the rate at which competence is obtained during case base construction. While deletion/editing policies are concerned with positive examples of problem solving, negative or failed retrievals also provide an opportunity for learning. In the introspective learning technique for learning feature weights described earlier, Bonzano et al. (1997) show that failure driven learning rather than success driven learning contributes most to the observed retrieval improvements. In the route planning domain, Fox and Leake (1995) use introspective refinement of index criteria, based on retrieval of cases that cannot be adapted, in order to include features that will lead to the retrieval of more adaptable cases in the future.

Leake and Wilson (1998) define three timing policies for case base maintenance: *Periodic*, *Conditional* and *Ad Hoc*. Periodic maintenance occurs at a regular interval with respect to the CBR cycle. For example, data collection may happen at the end of each problem solving cycle. Conditional timing occurs in response to a pre-defined condition, such as when the case-base

reaches a certain size. This is the policy used by Smyth and Keane's (1995) maintenance strategy. Ad Hoc timing occurs when maintenance is carried out in response to conditions outside the case base itself, such as when new cases are added from another data repository.

3.8 Interactive CBR

Up to now we have made the assumption that the target problem is fully specified before retrieval is initiated. However, in many situations information on the problem may be only partially available; in which case the objective of the CBR system is to incrementally guide the user into fully specifying the problem so that a solution can be obtained.

3.8.1 Incremental CBR (I-CBR)

Doyle and Cunningham (2000) describe an incremental question and answer approach which allows on-line users to quickly build up a suitable laptop configuration. The ordering of the questions in this approach is based on choosing features according to the information gain measure described earlier. This work is based on earlier work in the domain of electronic fault diagnosis (Cunningham & Smyth 1994) where it is difficult to gather a full case description in advance. In this type of diagnosis there is potentially a lot of information that could be used to solve the problem, however the goal is to elicit the minimum required for full diagnosis. This is particularly pertinent since some of the information is *expensive*, requiring labour intensive testing on the part of the diagnostician. Therefore, it is important to request expensive information only when it is determined that it will contribute to the diagnosis. While diagnostic tests can be carried out in any order, one of the drawbacks in using I-CBR in a domain where the user may not be an expert (such as with the on-line laptop guide) is that question ordering may appear arbitrary. The information theoretic criteria used to choose discriminating questions is insensitive to the intuition humans may have about what features are important. Moreover, such systems are generally not designed to allow the user to take the initiative and steer or conclude the dialogue.

3.8.2 Conversational CBR

The term Conversational CBR (CCBR) was introduced by Aha and Breslow (1997). It refers to CBR systems in which the user has an active part in the inferencing process. Like the incremental systems we have just described, CCBR are characterised by interactive dialogue in which the user is guided through a question and answering sequence during the retrieval phase. While the goal of an incremental system is to minimise dialogue length, CCBR systems are more concerned about facilitating the human subject – providing high dialogue quality and allowing the user to guide or to finish the dialogue at any time.

Unlike I-CBR systems in which the problem description is represented using attribute-value pairs, cases in CCBR systems are represented using a text description of the problem and a series of question and answer pairs. The NaCoDAE system of Aha et al. (2001) is a decision support system in which the user may initiate a dialog by entering a free text description of the problem. The system then retrieves a ranked set of cases by calculating the similarity of the text description in each case to the case description in the query. A ranked set of solutions is presented to the user, as well as a set of questions ranked according to the frequency with which they occur in the retrieval set. The user can sustain the dialogue by answering *any* of the questions posed. The system adds the question and answer to the query specification, and computes a new retrieval set. The interactive dialogue will continue until the user finds a solution they require. At any time, though, the user may terminate the dialogue by choosing one of the ranked solutions that is presented at each iteration. Research in CCBR concentrates mainly on authoring or refining the interactive case representations used with this technique (Aha & Breslow 1997) or on *dialog inferencing*, which involves ensuring that questions are not posed that have already been implicitly answered by the user (Aha et al. 1998).

3.8.3 Case Completion

Case completion (Kamp et al. 1996, Burkhard 1998, Lenz et al. 1998) in many respects addresses similar issues to I-CBR and CCBR. Case completion is the process of incrementally filling out a target case under the advice of a reasoning system like CBR. In the field of diagnosis, Burkhard (1998) proposes that the case representation has to be flexible enough to capture the differing diagnostic contexts in which a case may be used. He suggests that a case should not necessarily have homogenous structures or attribute value pairs since different parts of a case may be relevant for different problem contexts. Allocating a solution part to a case is not appropriate in certain contexts as a solution description may be a problem description. Because diagnosis is a process, case representation should be able to reflect the steps required to reach a satisfactory answer.

The key observation by Burkhard is that cases with heterogeneous representations will always have missing values with respect to one another. Thus a retrieval mechanism will have to be designed to handle flexible case representations and missing values.

Figure 3.8 illustrates the interactive CBR process in terms of the cycle we introduced at the start of the chapter. As we can see, the process of retrieval involves several iterations in which the user is actively involved. The retain stage is not illustrated here. In this type of CBR the retain phase is not engaged until a case has been specified to the satisfaction of the user.

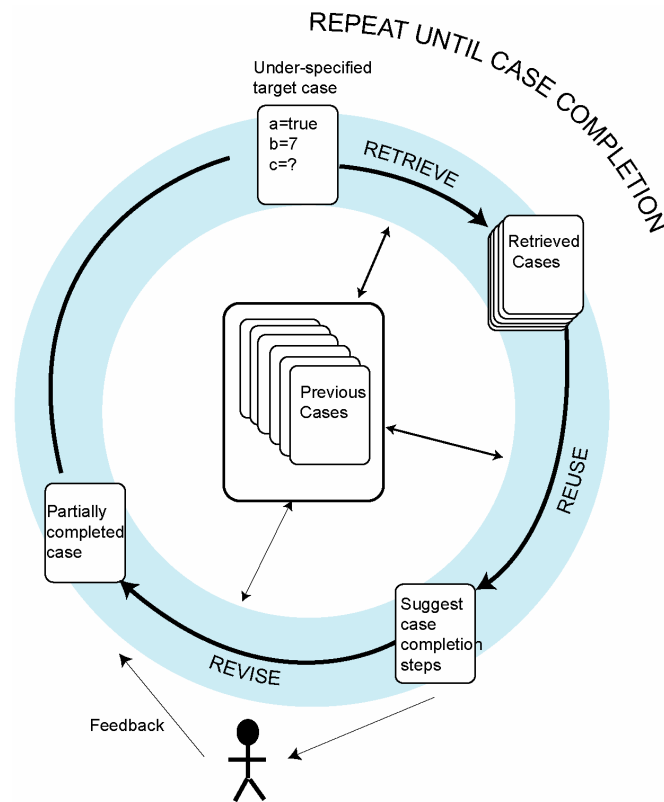


Figure 3.8: The interactive CBR process.

3.8.4 Case Retrieval Nets

We can view the k -D-tree described earlier as a top-down approach to retrieval in which retrieval proceeds from the root to the relevant leaves by testing the value of a query attribute at each node. This means that the value of the query attributes must be available in a certain order. If a particular value is missing it may cause retrieval to proceed down the wrong branch of the tree.

A Case Retrieval Net (CRN) is a memory structure based on the concept of *remembering as reconstruction*. This is a bottom-up approach to retrieval in which cases are indexed based on the information they have in common. A partially specified query may be used to retrieve relevant cases from a CRN based on the current knowledge that is shared between the cases. The components of the retrieved cases are candidates for completing or filling out the query case.

Lenz and Burkhard (1996) give three major conditions that should be met by a retrieval system, and which are fulfilled by the CRN memory structure:

Efficiency: Access to cases should avoid an exhaustive search through case memory.

Completeness: Every sufficiently relevant case should be found during retrieval.

Flexibility: There should be no inherent restrictions under which a piece of knowledge can be potentially recalled.

The basic unit of information in a CRN is an *information entity* (IE) which can represent an attribute value pair. For instance in a diagnostic situation, an IE can be used to represent a special symptom, a device parameter, a fault hypothesis, etc. (Kamp et al. 1996). A case is a chain of such

information entities. IEs are atomic information units which can be compared for different cases. Since the IEs associated with each case may overlap, a case base can be thought of as a network of IE nodes and case nodes.

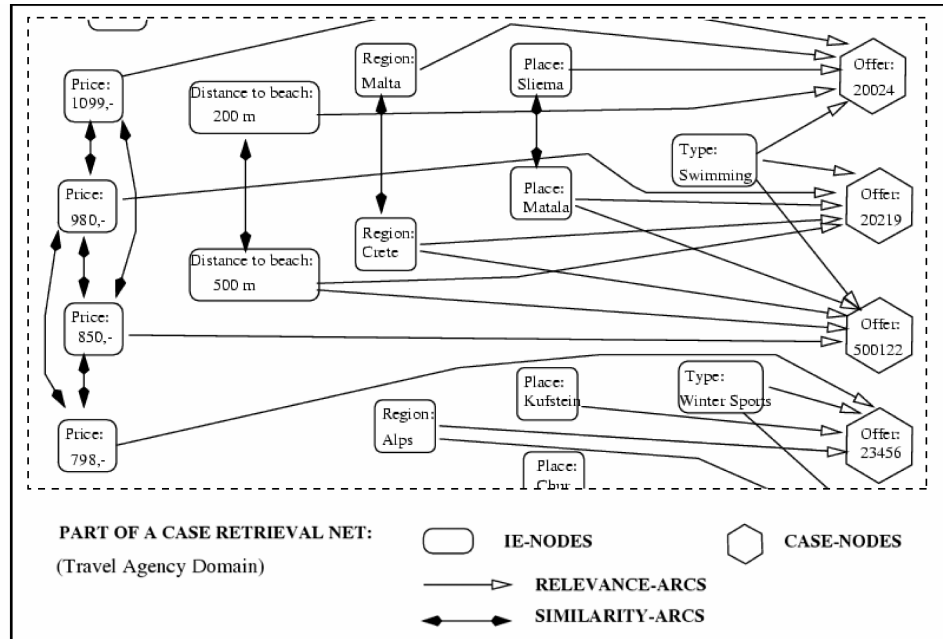


Figure 3.9: The CRN structure (from Lenz & Burkhard 1996)

Figure 3.9 illustrates the structure of a CRN using data from the travel case base (Lenz 1993). In this figure each case node is associated with a set of IE-nodes using a *relevance arc*. For example, the case node *offer 20219* is described by the following IE-nodes: Place: Matala; Region: Crete; Distance to Beach: 500m; Price: 980. We can see that several other cases also share these IE-nodes. Each relevance arc encodes the *amalgamation function* (described earlier) which calculates the global similarity of that case for a target query. Each IE-node is connected to other IE-nodes of the same type by means of a *similarity arc*. For instance, the IE-nodes describing *price* are all connected by similarity arcs. The similarity arcs encode the local similarity functions for a particular attribute. In the case of the price attribute the similarity function could be a simple distance measure used for all price IE-nodes.

CRN retrieval

CRN retrieval has three stages:

1. activating the IE-nodes described by the query description
2. propagating activation according to the similarity functions through linked IE-nodes
3. collecting the total activation for each case node.

One of the contributions of this thesis is that we demonstrate that a CRN memory structure can be used in Collaborative Filtering. In Chapter 6 we describe in detail the activation process of the CRN in that context.

The spreading activation technique used in CRN is similar to that used in Semantic Nets (Collins & Loftus 1988). Unlike semantic nets, the activation implies a *similarity* based connection, not a semantic connection, between activated entities. The propagation functions and the computation performed locally in each node indicate a connection with Neural Networks. However, the nodes in CRNs are symbolic, and the activation of each node has meaning, in contrast to the sub-symbolic representation of neural nets where patterns of activation are what matter.

3.9 Case-Based User Profiling

Up to this point we have described CBR systems in which the user plays an active part in the query formulation. In this section we describe systems in which the case query (a user profile) is automatically built by monitoring the user's behaviour, and where the system is adapted to take into account the information contained in the profile.

In the domain of Information Retrieval the potential for adaptive user modelling has long been recognised (Myaeng & Korfhage 1986, Rich 1989, Goker & McCluskey 1991). There are varying degrees of complexity with which we can model the user. At a very high level a user model encodes the knowledge available to the user, the goals and the plans of the user. Generally, in information systems, however, the user model is a much simpler affair containing a list of terms relevant to the user's information needs (Foltz & Dumais 1992). The list is usually short for querying tasks and longer for information filtering tasks/recommendation tasks. The objective of employing a user profile is to anticipate the user's information requirements, finding information he/she is interested in and removing items that might not be liked.

CBR, because of its similarity matching capabilities, has been employed as a recommendation engine to match case-based user profiles to items that the user may potentially like or find useful (Smyth & Cotter 1999b, Arslan & Ricci 2003). We review three systems in which case-based user profiling is used. In each of the three systems we see that the following are required.

- Content description for the items being recommended/filtered. (Vocabulary knowledge).
- A case representation format that captures the information need of the user and is compatible with the case representation used for domain items. (Case knowledge).
- A similarity model by which to measure the similarity between the user profile and the items being filtered/recommended. (Similarity knowledge).

3.9.1 PTV

The PTV system uses case-based and collaborative techniques to generate web-based television guides customised to the viewing tastes of each of its users (Smyth & Cotter 1999b). Smyth and Cotter argue that a personalised television guide becomes a necessity with the increasing amount of

content available through satellite and digital services. Programmes are represented as simple cases in the PTV system (see Table 3.2).

Table 3.2 An example of a programme case in PTV

ER	
Genre:	Medical Drama
Country:	USA
Language:	English
Cast:	Anthony Edward
..	..

Users can give the PTV system feedback on their favourite programmes, either by typing the title into a text box or by clicking on a rating icon beside a programme listing (see Figure 3.10). In this way, the user builds up a rating profile which can be used by PTV’s *collaborative* recommendation engine. However, in order to construct a case-based profile for the user, the system summarises the features associated with the programmes in the rating profile. This produces a case-like profile encoded using the same features as the programmes.

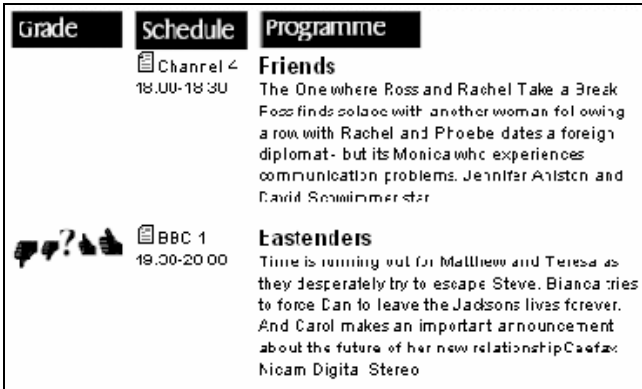


Figure 3.10: The PTV rating system

Therefore, similarity between a profile and a given programme can be calculated using the general similarity metric shown in Equation 3.4. In many respects the developers of PTV faced the same problems as we did regarding content mark-up. They suggest that the case-based approach in this domain is difficult because of the knowledge engineering required to develop suitable case representation and similarity metrics. In addition, they describe how recommendation using the case-based profiling tended to retrieve the same type of programmes, leading to reduced diversity. This is pertinent in that the programme content description available to the PTV researchers was quite limited. For instance, some programmes might only be characterised with a key word such as ‘comedy’ which is not sufficient to discriminate between younger viewers who watch ‘Friends’ and their parents who watch ‘One Foot in the Grave’ (Smyth et al. 1998). It is not surprising, therefore, that an evaluation of the system demonstrated that the collaborative approach out-performed the case-based approach.

3.9.2 CASPER

The CASPER (Case-Based Profiling for Electronic Recruitment) research project has as its objective the building of intelligent aides for use in specialised retrieval applications (Bradley et al. 2000). The test bed application for this project is an on-line recruitment site (<http://www.jobfinder.ie>). CASPER employs a two-stage retrieval strategy. The first stage requires the on-line job seeker to enter a description of his preferred job. This query is sent to the CASPER server where a similarity based ranking of jobs is returned. Each job is described as a case with a fixed set of features such as job type, salary, skills and experience. Similarity between the query and the job in the case base is calculated using Equation 3.4. The CASPER researchers have developed a domain ontology in order to model the similarity between symbolic features. Using concept trees, feature similarity is calculated using *subsumption* relationships and the distances between nodes. Figure 3.11a illustrates the subsumption relationship. A query specifying the feature value ‘OOP’ will match exactly with features with values that lie below the ‘OOP’ value on the same branch, such as ‘C++’ or ‘Java’. The distance relationship is shown by Figure 3.11b. This time, the feature value submitted in the query is ‘Java’. Based on the number of inter-nodal edges between concepts, ‘Pascal’ (distance = 4) is judged to be less similar to ‘Java’ than ‘C++’ which has a distance of 2.

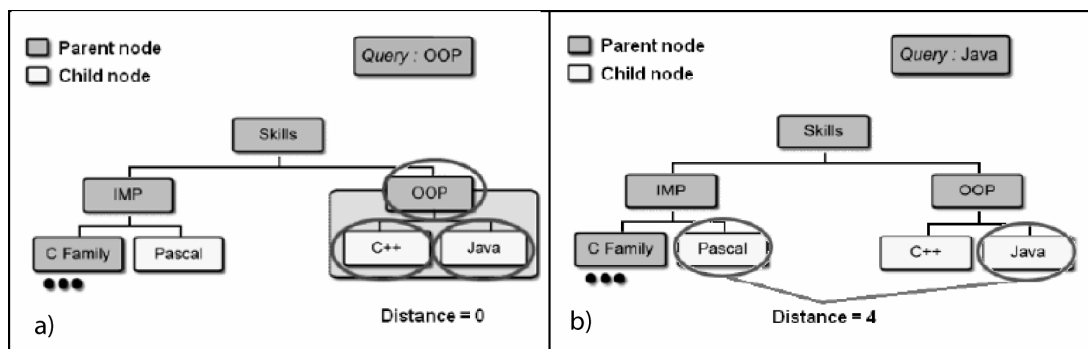


Figure 3.11: The concept trees used for similarity calculation in CASPER (from Bradley et al. 2000)

The first stage retrieval allows job cases to be ranked according to the similarity to the query case. CASPER’s second stage involves classifying the retrieved cases according to their relevance to a client-side stored profile. The CASPER researchers have developed several metrics to measure user interest in jobs such as time spent perusing a job description (Rafter et al. 2000, 2001). This has allowed them to associate a set of job descriptions with each user which can be classified as either relevant or irrelevant. These job cases are not generalised into a single user profile (as was suggested in the PTV project); rather, they are used as training data for a *k*-Nearest Neighbour classification of each job case returned by the first stage (see Figure 3.12). Thus the jobs retrieved by the user’s initial query are filtered to return only those that are pertinent to job descriptions the user has expressed interest in the past.

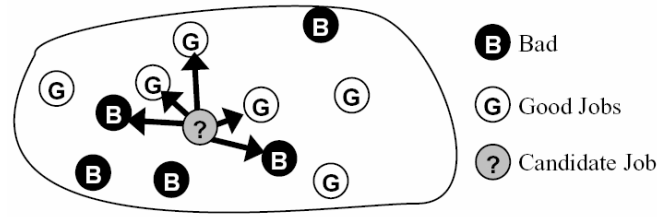


Figure 3.12: On the client side, CASPER uses k -NN to classify candidate jobs for the user (Bradley et al. 2000)

The client-side classification has the advantage of preserving privacy, particularly if this information can be standardised so that it can be applied to other information services.

However, this example does highlight the fact that CBR does come with a knowledge engineering expense such as the requirement to author a domain ontology. The Semantic Web project (Berners-Lee 1998), although in its infancy, has as its goal the development of the type of ontological information that will allow systems such as CASPER to become widespread in the future.

3.9.3 Dietorecs Travel Assistant

The Dietorecs Travel Assistant is a travel advisory system which can guide the user while he/she is putting together a full set of travel related activities (Arslan & Ricci 2003). The system uses a *mixed initiative* approach, similar to the conversational approach we described earlier. What is unusual about the Dietorecs approach is that advice on the possible travel configurations is given based on stored previous user sessions, each of which is stored as a case (see Figure 3.13).

As the user develops his/her travel plan on-line, the system iteratively computes similarity to other user sessions, and makes suggestions on how to complete the itinerary. Once the user's trip is agreed, the session is stored as a case for possible use by another on-line traveller.

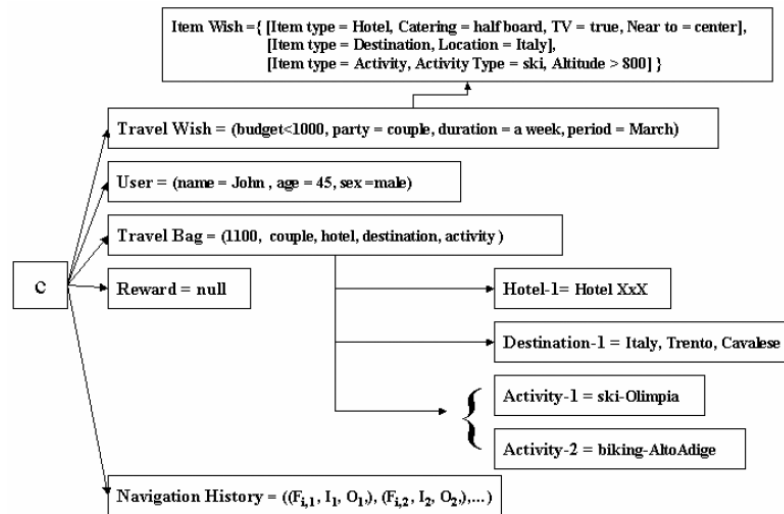


Figure 3.13: An example of a case in the Dietorecs travel assistance system

Dietorecs is an example of a system in which user configuration information is stored for other users to make use of. We will suggest that the playlist configurations in Smart Radio are another source of knowledge that can usefully be passed to nearest neighbours.

3.9.4 Perspective on CBR in the Music Domain

CBR's major advantage over rule-based or model-based systems is that the information contained in the case knowledge is not generalised to form rules, but instead lazily invoked at query time. However, if the development of the case and similarity knowledge is an intensive process, CBR has less of an advantage. In the music domain, our problem is fundamental: a lack of inexpensive vocabulary knowledge. Without this, case knowledge and similarity knowledge cannot be developed. The usual way of creating such vocabulary is for domain experts to establish the fundamental concepts, objects, relations, etc, which exist for a given domain, which is an expensive process.

However, it is implicitly acknowledged in the literature that there are varying degrees of strength of CBR. A *strong* CBR system makes use of a clear domain model to develop a case representation that may have surface features, which are used for indexing, and deeper, structural features which are used for similarity matching. This type of CBR is close to the cognitive model of CBR (Leake 1996). A *weak* CBR system, on the other hand, may calculate similarity between cases without reference to the semantics of the case representation. In this situation there are no structural features, or they are not easily available, and similarity is calculated based on the surface features available to the developer, using 'a one size fits all' metric such as the Euclidean or the City Block measure. Most applied CBR systems lie somewhere between the two, dependent on the difficulty of modelling the knowledge containers. In Chapter 5 we will describe our implementation of a weak CBR system to augment our ACF recommendation process.

3.10 Conclusion

In this chapter we reviewed case-based reasoning, an applied AI methodology which is suitable for problem solving in domains where it is difficult to elicit first principle rules. Unlike rule-based or model-based systems, CBR systems reason by drawing upon the knowledge stored in previously solved cases. While CBR generally has lower knowledge engineering overheads than systems that use first principle reasoning, it still relies upon the availability of four sources of knowledge: *vocabulary knowledge*, *case knowledge*, *similarity knowledge* and *adaptation knowledge*. CBR's big advantage over rule-based systems is that the *case knowledge* is deployed at query time, and does not have to be generalised *a priori*. However, if the other sources of knowledge are not available, CBR systems are forced to confront the knowledge acquisition bottleneck. For instance, the PTV system, like the Smart Radio system, has problems acquiring vocabulary knowledge – the basic attributes and relations within a domain that allow useful cases and similarity metrics to be developed. The CASPER system faces a knowledge bottleneck in developing the similarity

knowledge required in its concept trees. We described how the *Reuse* phase (which uses adaptation knowledge) of the CBR cycle also poses serious problems for general CBR development since it forces developers to acquire some measure of first principles reasoning, which CBR systems are supposed to circumvent. With CBR increasingly being deployed in domains with large repositories of data, we discussed the scalability issues associated with lazy reasoning. While *k*-D-trees are efficient in terms of retrieval, they are problematic in domains in which only a partial problem description may be available. An example of this is interactive CBR, in which a user may only have a partial query and may need help in honing in on the solution or configuration he/she requires. In this situation, the reasoning system engages the user in a dialogue in order to guide him/her towards a solution. We reviewed CRNs, an efficient memory structure suited to retrieval in situations where queries are incomplete. We revisit Case Retrieval Nets later in this thesis where we apply them as a memory model for collaborative filtering techniques. CBR is increasingly being used on the Internet as a reasoning agent for product selection, configuration or personalisation. For example, CBR user profiles are used to represent items that users have liked/disliked in the past in order to make predictions on things they may like or dislike in the future. The PTV system builds a generalised case-profile from features of programmes that users have rated. Thus, the profile can be matched against new programmes appearing in the TV schedule. As mentioned earlier, one of the problems of the case-based approach in this domain is that good content description is hard to obtain. The CASPER system does not generalise the job description cases used to represent a user's job interests. Instead these instances are used to classify retrieved job descriptions as relevant or irrelevant using the *k*-NN algorithm. The Dietorecs system does not use user profiling, as such, but employs a case base containing records of other users' travel plan configurations, in order to interactively advise the user of possible itinerary scenarios.

While CBR is a good example of a 'content-based' personalisation strategy in which descriptions with clear semantics are used, we suggest that there are varying strengths of CBR. How case and similarity knowledge is implemented in a CBR system is dependent on the domain knowledge that is available. The next Chapter introduces Automated Collaborative Filtering, a technique which is often employed when domain knowledge is not readily available.

Chapter 4: Automated Collaborative Filtering

Automated Collaborative Filtering (ACF) has emerged as a popular recommendation strategy for on-line systems (Schafer et al. 1999). ACF does not attempt to model any aspect of cognitive psychology as Case-Based Reasoning (CBR) 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 ‘content-less’ approach to filtering or recommendation (Resnick et al. 1994, Balabanovic & Shoham 1997).

While a content-based technique like CBR is used to offset the knowledge elicitation bottleneck of a rule-based or model-based system, it still relies upon having the basic vocabulary that allows cases and similarity measures to be defined. In some domains this knowledge is not readily available. ACF can be used as a recommendation strategy in such scenarios. Unlike CBR cases, ACF profiles are marked up, not with a set of descriptors, but with a set of scores. Each score is a subjective rating by a user of the item in question. These ratings can be explicitly submitted or derived implicitly by monitoring the user’s usage of the item. The knowledge engineering task for such a representation is generally low since it is collected from a set of distributed users, and can be gathered as part of the functionality of an already running system. For example, users could be provided with a facility to rate items suggested by a content-based recommender. This would give the type of ‘content-less’ representation shown in Table 4.1

Table 4.1: A ‘content-less’ representation

	Love Me Tender
User A	0.2
User B	0.8
User C	-
User D	0.2

Table 4.2: A case-like representation

Song:	Love Me Tender
Title:	Love me Tender
Artist:	Elvis Presley
Type:	Male solo
Genre:	Rock’n’Roll, Pop
Style:	Ballad
Album:	Elvis Love Songs
Date:	1956
Instrument:	vocal, acoustic guitar
Keywords:	love, romance
Tempo:	slow
Film:	Love me Tender 1956

While the description of the song in Table 4.1 is not as immediately interpretable as the content-based description in Table 4.2, the description is far from ‘content-less’. The rating data implicitly contain meaning with respect to how the song is considered among a population of users. It is this implicit knowledge which is leveraged by a technique like ACF to make recommendations. With

rating data, ACF makes use of the subtle distinctions people make when selecting or rejecting from any set of items. This type of knowledge (aesthetic criteria, emotionally or culturally based responses etc) is very hard to encode into content-based systems.

In the example in Table 4.1 we show one column of the user-item matrix which demonstrates that items can be described in terms of user scores. Likewise, were we to show more columns we would see that each user can be profiled in terms of the items they have rated (see Table 4.4). Although often proposed in the literature as very different types of techniques, we will show in Chapter 6 that ACF and a content-based strategy such as CBR are very similar in terms of methodology. We will also show how techniques used in CBR can be applied to ACF.

Research in collaborative filtering draws from work in several different fields. Papers cited in this chapter describe research topics from information retrieval, distributed artificial intelligence, machine learning, statistics, computer-human interaction, user-modelling and social choice theory.

4.1.1 Early Collaborative Filtering

Collaborative filtering systems emerged out of early research into the problem of information overload in a networked environment (Denning 1982, Palme 1984, Hill & Turoff 1985). Malone identified three types of intelligent information filter that might allow people to find desired information and eliminate undesirable material: cognitive, social and economic (Malone et al. 1987). The categories are based on the information resources that each filter draws upon to make a prediction on an item for a particular user. Cognitive filtering systems analyse the text within documents and match the results against a user-profile. These types of systems are generally now known as content-based or knowledge-based filters and are dealt with in more detail in Section 4.3. Economic filters select documents based on the costs and benefits of producing and reading them, a criterion which will become relevant as content on the Internet increasingly requires payment. Social filtering, according to Malone, selects documents based on the evaluation of other people with whom the target user may have a relationship. A social filtering system recognises that people may choose to read material annotated by someone in authority or by a trusted source.

In its purest sense, collaborative filtering refers to a process of filtering solely according to the opinions of other users (Resnick et al. 1994). The first collaborative filtering systems, however, were essentially content-based systems augmented by some characteristics of social filtering. The Lens system of Malone et al. was primarily a rule-based content filter for e-mail that included a social filtering component which allowed the receiver to set his/her filter to receive messages according to the characteristics of the sender (Malone et al. 1987). The first social filtering system that involved a collaborative element was Tapestry, a mail filtering system developed at the Xerox Palo Alto Research Center (Goldberg et al. 1992). Tapestry allowed members of its user base to annotate each document with details on how interesting (or uninteresting) the reader had found the document. The document contents and the annotations could be accessed by each user's filter. This allowed users to filter the system for documents on a topic that had been annotated by a particular

person, or to receive documents ‘replied to by user A AND user B’. The Tapestry system had some drawbacks. Users needed to learn a query language for the system, TQL, and were required to actively annotate their documents for the system to work effectively. Furthermore, since it was assumed that users worked in a relatively small environment it provided no means of automatically identifying users with similar interests. Maltz and Ehrlich employed a similar approach using the term active collaborative filtering to describe a process of providing ‘pointers’ to recommended documents for their friends and colleagues (Maltz & Erlich 1995).

4.2 Filtering vs. Recommending

Early collaborative filtering technology was described in terms of existing filtering techniques used by the information retrieval community. More recently it has been referred to as a ‘recommender’ technology (Schafer et al. 1999). Are filtering and recommendation the same thing? To answer the question it is useful to look at the distinction drawn between filtering systems and information retrieval systems. Belkin and Croft (1992) have defined three primary differences between the information filtering task and information retrieval task. Firstly, a user-profile in information filtering represents the user’s interests over a period of time, while query terms in an information retrieval request represent short-term information requirements which can be fulfilled by the retrieval process. Secondly, information filtering is typically used where there are streams of incoming data, while information retrieval involves searching in a structured database for information to match the query terms. In this sense filtering is passive, while retrieval is active. Thirdly, information filtering involves removing items from an incoming stream whereas information retrieval is concerned with finding information.

The recommendation task has aspects of both filtering and information retrieval. A user profile in a recommendation context represents the user’s interests gathered over time. User profiles can capture long-term or short-term interests, and are generated by analysing the user’s past use of the system resources. Rather than being a passive activity we can view the recommendation task as a type of automated search strategy where the goal is to anticipate the user’s future requests based on the knowledge contained in the user-profile.

We have previously suggested that the recommendation task has not yet been adequately defined which has implications for how it can be evaluated (Hayes et al. 2002b).

Table 4.3: Comparison of filtering, IR and recommendation tasks

Filtering	Information Retrieval	Recommendation
User profile represents user interests collected over time.	Query terms represent immediate information requirements.	User profile represents user interests collected over time.
Passive	Active	Active
Removal of unsuitable material.	Finding material that matches query terms.	Finding material that matches query terms. Removal of unsuitable material.

4.3 Filtering Basics

Towards the end of the previous chapter we described some case-based personalisation systems in which items were recommended based on the user-profiles built up by each system. As we suggested in the last section, the recommendation task and the filtering task can be viewed as similar processes. Since ACF was originally developed as a filtering methodology, in this section we review some of the basic ideas behind filtering.

The fundamental idea behind filtering involves the removal of unwanted items from an incoming stream of items and the preservation of items that would be of interest to a target user. The most obvious way to do this would be to develop an explicit representation of the incoming items, and then attempt to match these against a user profile (Malone et al. 1987, Foltz & Dumais 1992). The terms of the user profile might be explicitly articulated by the user as in the Tapestry system (Goldberg et al. 1992) or learned by the system (Lang 1995, Pazzani et al. 1996).

Figure 4.1 illustrates a recommendation scenario whereby the system learns the terms of the user-profile by monitoring user feedback (Pazzani et al. 1996).

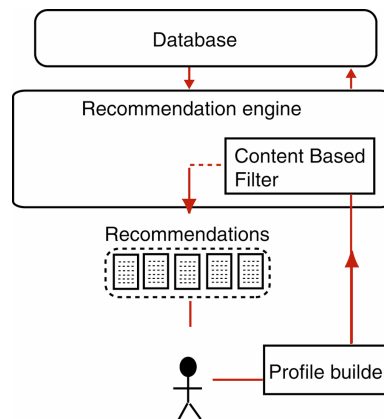


Figure 4.1: A content-based filter

The strength of the content-based filter is that it will consistently find new content that matches features in the user profile. The drawbacks to this type of filter come down to the issue of knowledge elicitation and representation. Burke suggests that a distinction should be drawn between content-based filters and knowledge-based filters depending on the degree of knowledge engineering required for either (Burke 1999, 2000). According to this definition content-based filters have a lower knowledge authoring overhead than knowledge-based filters. Content-based systems tend to operate in open environments in which the feature extraction process is automated. An example would be the representation of a web page as a vector of keyword terms (Salton et al. 1975). However, automated feature extraction tends to offer only a shallow analysis of the content items, ignoring aesthetic and qualitative properties. The similarity metrics used in content-based systems tend to be relatively simple, 'one size fits all' measures such as the cosine measure (Salton et al. 1975). A content filter can learn a user profile by recording the features of items the user has

rated in the past. As such, content-based filters suffer from a tendency to only offer the same type of content that a user has previously used (Pazzani et al. 1996, Balabanovic & Shoham 1997, Smyth & Cotter 1999b). Since the content representation employed is simple (perhaps a set of keywords as described by Foltz & Dumais 1992), a content filter cannot infer new types of content beyond those described in the user profile. Furthermore content filters cannot distinguish between what the active user would consider to be items of low or high quality. For instance, in the PTV television programme recommender system, the content-based filter cannot distinguish between programmes marked up as ‘comedy’ that are not particularly funny to sections of the viewing public (Smyth et al. 1998). Crucially, whereas text is readily amenable to parsing by a computer, useful features cannot be extracted from some items, such as audio or visual material, without considerable computational expense and/or expert knowledge (Tzanetakis et al. 2000, Grimaldi et al. 2003).

Knowledge-based filters encapsulate ‘expensive’ information such as the fundamental concepts, objects, relations, etc, which exist between content items. This type of ontological information needs to be marked up by hand, perhaps with the aid of knowledge discovery techniques (Fayyad et al. 1996). Likewise, knowledge-based similarity metrics are more sophisticated affairs, taking into account the more elaborate representation of the content (Burke 2000). Whereas knowledge-based filters should be able to overcome some of the problems ascribed to content-based filtering, the amount of knowledge engineering required, particularly in a dynamic on-line system, makes true knowledge-based filtering extremely difficult (Clerkin et al. 2001a). The many issues still surrounding the development of the Semantic Web project (Berners-Lee et al. 2001) are testament to this. This brief discussion is sufficient to introduce the filtering techniques used by ACF which on face value appear quite dissimilar to ‘content-based’ filtering.

4.4 Principles of ACF

4.4.1 Introductory Ideas

The basic assumption of the ACF is that, rather than relying upon content descriptions as a basis for filtering, it may be easier to harness the decisions made by other like-minded users in accepting or rejecting the system assets. Unlike the ‘content-based’ filter, a collaborative filter requires access to the usage data of other users of the system. It does not require that the system assets be modelled in any way, which is an enormous advantage where the representation task is onerous. The basic idea behind ACF can be illustrated in Figure 4.2 where all three users have expressed an interest in assets A, B & C. (For instance they may have listened to songs A, B & C.) The high degree of overlap indicates that these users may have a shared taste in music. Further it seems safe to recommend assets D and E to user 1 because they have been ‘endorsed’ by Users 2 and 3. However, as we will discuss, the task of assembling a group of neighbours to create a filter that

approximates the target user's interests requires a lot of use-data, since the degree of intersection between individual users varies considerably.

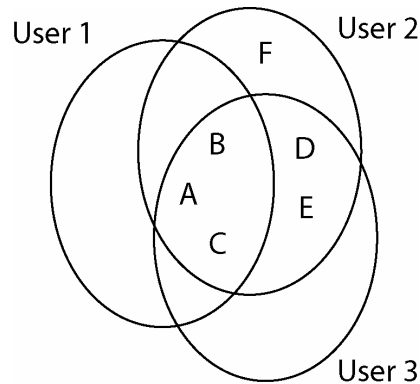


Figure 4.2: ACF takes advantage of the overlap in interests within communities of users

ACF takes advantage of the overlap in interests within communities of users to develop a filter for each user based on a neighbourhood of ‘like-minded’ users. The key assumption on which any ACF algorithm rests is that human preferences can be correlated, and that informed prediction can be made based on these correlations. Until recently, the ACF algorithm did not have formal roots in any discipline as CBR does in cognitive science. Despite its increasing application within on-line commercial systems, it has often been viewed as ‘the orphan child’ of applied AI, whose parentage is somewhat questionable (Billsus and Pazzani 1998). However, Pennock et al. have demonstrated that ACF techniques rely upon similar properties of determining and aggregating preferences that social choice theorists have been formally analysing for decades (Pennock et al. 2000). Using the conditions laid down by such theorists they show that the weighted, nearest neighbour implementation of the ACF algorithm is the only possible form of prediction function possible. In Section 4.5.2 we will discuss this implementation, as well as other techniques established outside the formalism of social theory.

4.4.2 Data Representation

Relevance feedback is a commonly accepted technique of improving retrieval quality in information retrieval (IR) systems (Rocchio 1971, Salton & Buckley 1990). Using relevance feedback, a user can indicate to an IR system which portions of a retrieved document set are useful to him or her. The IR system can reformulate the original query based on the feedback and return an improved retrieval set. This type of feedback is sometimes referred to as “querying by instance”. Unlike IR systems which rely upon content representation, collaborative filtering systems use relevance feedback as the only means of filtering or recommending.

The amount of data required depends to some extent on the type of data available. In this context, there are two distinct approaches to the ACF idea that can be termed explicit and implicit (Nichols 1997, Oard & Kim 1998, Claypool et al. 2001, Rafter et al. 2000, Hayes et al. 2001,

O’Sullivan et al. 2002). With the explicit approach the user is asked to rate assets. Such an approach was taken by the GroupLens project where users are asked to rate Usenet articles (Konstan et al. 1997). However, asking users to rate articles imposes a cognitive load that users may refuse. Grudin has observed that users may not participate in explicit grading schemes if they are not aware of the benefits to them (Grudin 1994). Thus explicit datasets may be sparse because of the work required by users (Oard & Kim 1998). Consequently there are several strands of research on gathering implicit data without imposing a feedback obligation on users (Morita & Shinoda 1994, Konstan et al. 1997, Lieberman 1997, Rafter et al. 2000). For instance, implicit data points might represent individual page impressions at a website. Implicit data contains less information and can be noisy in the sense that users might not like some of the items they have used. This can be seen in Table 4.5, which is an implicit version of Table 4.4. The information that User 2 dislikes asset D is lost in the implicit approach. Because of this data noise and loss of information, more data is needed to produce good recommendations with a purely implicit approach. However, it can be used to supplement a sparse dataset based on explicit ratings (Konstan et al. 1997, Rafter et al. 2000). Research has been done on deriving more precise implicit values based on a qualitative measurement of specific user actions. For instance, Morita & Shinoda found that the time people spend reading a Usenet article is correlated with their interest in it, but that there was no correlation between message length and reading time (Morita and Shinoda 1994). In the CASPER job finder system Rafter & Smyth use reading duration and the number of revisits to a job description as implicit metrics (Rafter & Smyth 2000).

Several similar implicit techniques have been applied by researchers (Lieberman 1997, Konstan et al. 1997, Mobasher et al. 2000a). In Chapter 6 we will discuss how data is captured and represented in the Smart Radio system.

Table 4.4: Data for use in ACF where users have explicitly rated assets

	A	B	C	D	E	F	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
User 5	0.6	0.6	0.8	?	?	0.7	?
User 6	?	0.8	0.8	0.7	0.7	?	?
User 7	0.7	0.5	?	?	0.7	?	?
User 8	?	?	?	?	0.7	0.7	0.8

Table 4.5: ACF data where users have not explicitly rated assets

	A	B	C	D	E	F	G
User 1	1	1	1	?	?	1	1
User 2	?	1	1	1	1	?	?
User 3	1	1	1	1	?	1	1
User 4	?	?	?	?	1	1	1
User 5	1	1	1	?	?	1	?
User 6	?	1	1	1	1	?	?
User 7	1	1	?	?	1	?	?
User 8	?	?	?	?	1	1	1

Irrespective of how it is collected, ACF data is usually sparse in that a user typically rates only a small portion of the items in the system (Sarwar et al. 1998). Equation 4.1 defines a metric for sparsity which is termed sparsity level.

Equation 4.1

$$\text{sparsity level} = 1 - \frac{\text{non zero entries}}{\text{total entries}}$$

The degree of data sparsity is domain dependent, but for sites with a large product range and many customers, the sparsity level can be in the range of 90–97%. For instance, the sparsity level of the MovieLens ACF dataset is 0.9369 (Sarwar et al. 2000b).

4.4.3 ACF as a Classification Task

In the user–item tables above, the missing values are indicated by a ‘?’. The prediction task has been viewed as an elaboration, or filling out of the sparse user–item matrix (Billsus and Pazzani 1999), but within the constraints of a real time system. We can view this prediction task as a classification or regression problem. In a situation where users have rated assets, the recommendation problem may be cast as the prediction of these ratings – a regression problem. Alternatively, it may be viewed as a classification problem – the classification being whether an asset will be liked or disliked. The measure of accuracy may be: 0/1 error for classification, absolute mean error (MAE) or root-mean-square (RMS) error for regression or a measure of the correlation of predicted ratings with actual ratings (e.g. Pearson's correlation coefficient). This allows for the type of evaluation that is common in machine learning where the data is partitioned into training and test data – using the training data to produce predictions for the test data. These estimates may be improved by using *k*-fold cross validation or by using a leave-one-out evaluation, i.e. a rating is predicted by using all the data except the rating itself (Moore & Lee 1994).

In Chapter 7 we will return to the issue of evaluation and suggest that the recommendation task is not as well understood as a classification/regression task, and that the goal of filling in the missing values in the user–item matrix does not take into account the active user’s current ‘use-context’.

4.5 Types of ACF

In machine learning terms a predictive algorithm can be described as being within a spectrum that ranges from being lazy to being eager. Both types of algorithms seek to approximate a function that can predict the outcome of new (query) data based on a set of training data. Eager systems generalise beyond the training data before observing a new query instance, building a global predictive model at training time. Lazy learners, however, do not build a model during training. These algorithms consider the query instance at query time and use the training data to approximate

the predictive function using a number of local functions based on instances in the training data (Mitchell 1997). Eager algorithms include decision trees (Quinlan 1993), back-propagation networks (Chauvin & Rumelhart 1995) and Bayesian belief networks (Heckerman et al. 1995). Lazy learning algorithms include the k -Nearest Neighbour algorithm (Aha et al. 1991) and CBR (Kolodner 1993).

ACF algorithms can be divided along similar lines into model-based algorithms that are eager and memory-based algorithms that are lazy (Breese et al. 1998). Both types of algorithm, however, seek to approximate a function that can predict the outcome for new data for a particular user. We will use the term *user-filter function* to describe this.

Pure model-based algorithms for ACF include the Bayesian network model used by Breese et al. (1998) and the back-propagation network used by Billsus and Pazzani (1999). Memory-based algorithms rely upon the retrieval of training data from memory in order to build a predictor at query time or close to query time. Hybrid approaches exist which are essentially lazy in spirit (partially lazy). These use eager, unsupervised learning techniques to cluster users based on items they have viewed, or to cluster items based on which users have viewed them (Breese et al. 1998, Ungar & Foster 1998a, Clerkin, Hayes & Cunningham 2001). We will now describe examples of model-based algorithms, but then focus more on lazy and partially lazy memory-based algorithms. Since this is the type of ACF used in Smart Radio, we will discuss memory-based structures in more detail and provide arguments for using them instead of a purely model-based alternative.

4.5.1 Model-Based Algorithms

Experiments with model-based algorithms have generally been performed off-line, and have the disadvantage of not taking into account an incoming stream of user feedback. The efficacy of these techniques in a real-life system has yet to be evaluated. Model-based algorithms tend to be probabilistic or based on machine learning techniques.

Probabilistic models

Probabilistic algorithms view the ACF task as one of establishing the conditional probability of an item having a particular rating score based on the ratings data collected to date (Breese et al. 1998, Miyahara & Pazzani 2000, Condliff et al. 1999). Breese et al. (1998) apply an algorithm for learning a Bayesian belief network (Chickering et al. 1997) from the ACF data in which each item has a set of parents that are the best predictors for its score. A decision tree represents the conditional probability table for each node. Figure 4.3 illustrates an example tree that encodes the probability that a user will have listened to the artist Leonard Cohen. In this example the parent artists are Bob Dylan and Joni Mitchell. The bars at the bottom indicate the probabilities of a user having listened and having not listened to Leonard Cohen, based on listening to the parent artists.

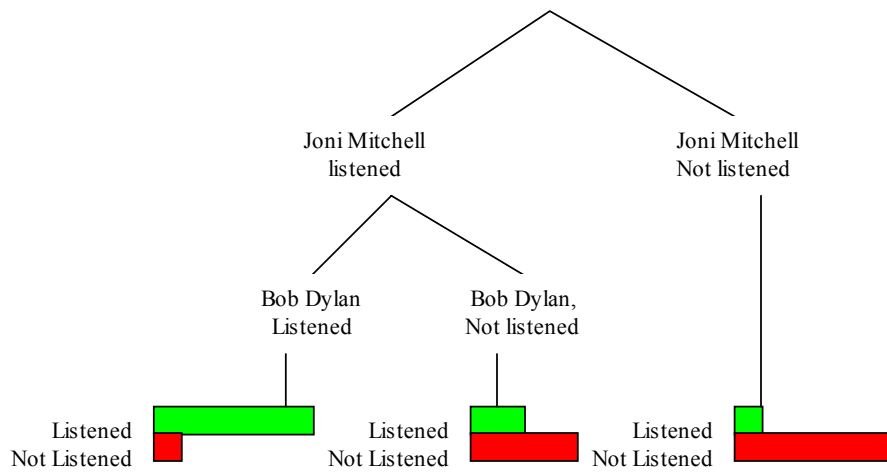


Figure 4.3: A decision tree for whether a user has listened to the artist Leonard Cohen

The Bayesian network technique was demonstrated to work as well or slightly better than correlation-based methods on some datasets. However, training the network proved to be computationally expensive (Breese et al. 1998). Thus, in a dynamic environment, this technique would be slow to adapt to the new data submitted by on-line users.

Popescul et al. (2001) and Schein et al. (2002) have recently proposed probabilistic models for ACF which combine content and collaborative features by using the expectation maximisation algorithm (EM) to fit the model to the data.

Machine learning models

Up to this point we have considered ACF as a technique of symbolic AI, where there is a clear correspondence between the semantics of the symbols being manipulated by an intelligent system and objects in the real or ‘physical’ world (Newell & Simon 1976). Billsus and Pazzani, however, adopt a sub-symbolic approach to learning a collaborative filter (Billsus & Pazzani 1999). They argue that the ACF process can be viewed as either a classification or regression task and that artificial neural networks provide the most flexible solution to either task. Prior to training the network they transform the data into a Boolean matrix on which they then run a dimensionality reduction process. Since dimensionality reduction is a key issue in memory-based algorithms, it is discussed further in Section 4.6.3. Making a prediction involves firstly converting the item’s user ratings into a Boolean vector, and then scaling the vector to the reduced dimension space. The resulting vector is fed to the trained artificial neural network in order to compute a prediction. Billsus and Pazzani’s predictive mechanism is a batch operation predicting all the missing values in the user profile at once. Experimental results from the Each Movie dataset demonstrate that dimensionality reduction allied with a Neural Net performed better than a typical memory-based technique in terms of overall predictive accuracy and precision at the top n ranked items. However,

the authors admit that it remains to be seen whether these techniques could be used to compute predictions in a live system under real time conditions.

Clustering algorithms

The objective of the clustering algorithm is the unsupervised classification of a set of objects. The term unsupervised refers to the fact that there are no *a priori* target classes used during training. Probabilistic clustering (Breese et al. 1998, Ungar & Foster 1998a,b) and machine learning based clustering (Kohrs & Merialdo 1999, O'Connor & Herlocker 1999, Clerkin et al. 2001a,b) have been performed on ACF data. Although these techniques are all eager, discussion is reserved for the section on memory-based algorithms, as clustering algorithms in ACF are generally used to augment an essentially lazy ACF algorithm.

Item-based ACF

Item-based ACF is a new technique introduced by the researchers behind the GroupLens project in an attempt to address the scalability issue when using k -nearest neighbour ACF in large sites. The key idea is to eagerly analyse the user-item matrix to identify relations between different items, rather than users, and then use these relations to determine a prediction for a particular user-item pair based on items the user has liked in the past (Sarwar et al. 2001).

4.5.2 Memory-Based Algorithms

Memory-based Algorithms in ACF approximate the user-filter function by combining the preferences of similar users. The approximation is carried out at query time (lazy) or near query time (partially-lazy). Whether the data available is binary or contains an explicit rating, the basic structure of the prediction process consists of a cycle of four distinct phases (Hayes, Cunningham & Smyth 2001).

Firstly, a neighbourhood for each user must be determined. The aggregated preferences of these users will form the basis of the predictive ACF filter. Determining the neighbourhood of users requires a metric for grouping or clustering similar users. If our task is to actively produce recommendations, rather than filtering an incoming stream, the next task is the extraction of candidate items from the neighbourhood use-data. Once candidate items have been extracted the next phase proceeds. The candidate items can be viewed as an incoming stream of items to be rated by the system, and shown to the user if positively scored. The preferences of the nearest neighbours are aggregated to make a prediction for each target item. In order to drive the ACF process, feedback must be elicited from the user either by inviting explicit ratings or by deriving ratings implicitly. At this stage the cycle returns to its starting point, and another iteration begins.

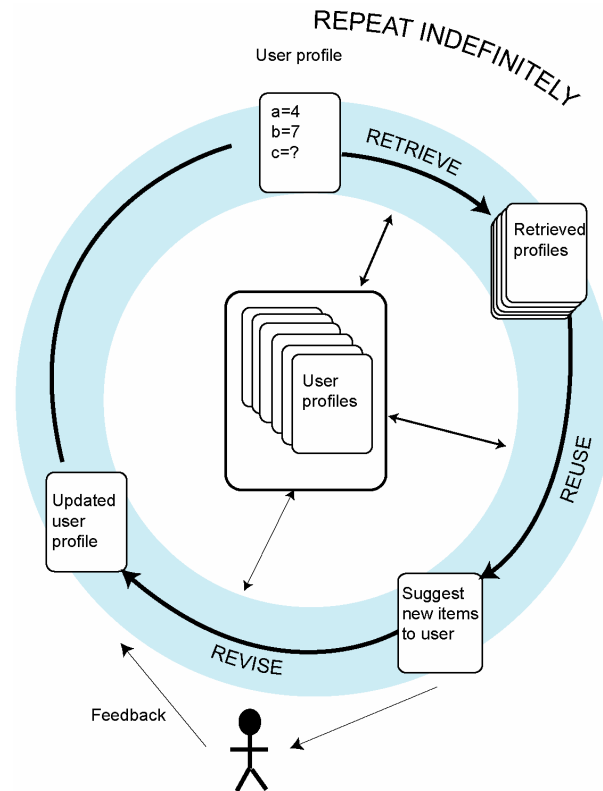


Figure 4.4: ACF is a cyclical, interactive process

4.6 Algorithmic Choice Issues

The argument as to whether to use a model or memory-based approach in ACF can be addressed by looking at the tradeoffs exhibited by eager and lazy learning algorithms. Lazy algorithms defer processing of their training data until query time when they combine the data to provide an answer. They thus have much lower training costs (sometimes zero cost) than eager algorithms. However, they typically have much greater storage and data maintenance requirements and have higher computational costs at query time. Eager algorithms, on the other hand, build an intentional concept description at training time and then answer queries using this description. While computational complexity may be lower at query time, eager algorithms can have very high training costs. This has implications for learning in a dynamic environment. Eager algorithms may not be able to learn very quickly from new training data, because of the cost in building a new concept model. On the other hand, lazy algorithms can use new data immediately by combining it with existing data to produce solutions that were not previously available.

The choice of algorithm for an ACF system is domain dependent and depends on the level of overall activity of resources and users within the system. For a system where user preferences and system assets change slowly with respect to the training period, a scheduled, eager process such as a Bayesian belief network may suffice. However, such a model would not be suitable for an environment in which user preference models must be updated quickly or frequently (Sarwar et al. 2001). In a time-sensitive domain such as the filtering of Usenet articles (Resnick et al. 1994) the

training delay would be unacceptable. A lazy approach will suit a domain with a high level of user feedback, new resources and a user expectation of prompt learning (Aha et al. 1991.) For instance, the Smart Radio system provides recommendations of new playlists compiled by member users. The system must quickly learn to filter unsuitable material from each user's set of recommendations or it will simply not be used.

4.6.1 Lazy and Partially Lazy

Previously, ACF has used lazy, memory-based techniques (Shardanand & Maes 1995, Resnick et al. 1994). The complexity of the correlation calculation alone for an $m \times n$ user-item matrix is $O(m^2 \cdot n)$, and it is this calculation which has proven to be the bottleneck for memory-based systems (Herlocker et al. 1999). With ACF being used in large sites such as Amazon.com, memory-based algorithms need to meet the constraints imposed by normal request response times. In some cases the neighbours that are selected to meet a request are not necessarily the best, but rather the most similar that could be found by sampling profiles in high speed memory (Herlocker et al. 2000). In fact, many lazy learning algorithms make expedient trade-offs in order to reduce computational complexity at query time. These algorithms are termed *partially lazy* (Aha 1997). Likewise many ACF implementations are partially-lazy or are augmented by data derived by eager algorithms. Three techniques proposed to augment a Lazy ACF process are similarity indexing, dimensionality reduction and pre-clustering users.

4.6.2 Similarity Indexing

Similarity indexing may be used to improve query-time performance using a batch calculation of user correlations. A correlations database of pair-wise similarities can then be maintained and used to identify neighbours and build recommendations at query-time (Konstan et al. 1997). In ACF terminology, memory-based techniques are often referred to as k -nearest neighbour methods. Little work, however, has been carried out on how techniques for improving the scalability of instance-based learning systems could be applied to ACF. The problem may lie within the nature of the data, which has missing values, and no class labels assigned *a priori* to each instance. For example, the condensed nearest neighbour algorithm (Hart 1968) requires class labels to produce a reduced set of learning instances. Similar problems apply to techniques for efficient memory indexing of instances. Techniques such as the k -D-tree, where instances are stored at the leaves of a tree with similar instances stored at the same or at nearby nodes (Bentley 1975, Friedman et al. 1977), may be impossible to implement where there are so many missing attribute values. One instance-based indexing technique that has proven scalable for large amounts of CBR data and is not sensitive to missing values is the Case Retrieval Net (CRN) (Lenz 1999). In Chapter 6 we describe our implementation of the ACF memory structure as a Case Retrieval Net. Hayes et al. (2001) provide a discussion of the advantages of the CRN as a memory structure for the ACF algorithm.

4.6.3 Dimensionality Reduction

Since query-time expense increases with the number of users and the number of items, reducing the dimensionality of the user–item matrix should lead to better query-time performance. If dimensionality reduction is employed to reduce the dimensions of the item rows, the effect is to produce a less sparse matrix, thus improving the chances of correlating similar users, by virtue of the fact that ‘similar’ items are now represented under a single dimension (Billsus and Pazzani 1999). One proposed technique is to use Singular Value Decomposition (SVD), a commonly used matrix factorisation technique, to reduce the dimensionality of the user–item matrix (Sarwar et al. 2000a, Billsus & Pazzani 1999). SVD is the basis of Latent Semantic Indexing (LSI), a technique used in information retrieval to solve the problems of synonymy and polysemy in a corpus of documents (Deerwester et al. 1990, Berry et al. 1995). Sarwar et al. (2000a) found that dimensionality reduction does reduce query-time for the calculation of neighbour users, but with mixed results for predictive accuracy. Billsus and Pazzani (1999) out-performed memory-based techniques in terms of predictive accuracy using dimensionality reduction techniques and a neural network classifier. However, determining the number of dimensions that is large enough to capture all the latent relationships in the dataset yet small enough to avoid over-fitting errors requires experimental evaluation (Deerwester et al. 1990, Berry et al. 1995, Sarwar et al. 2000a). Moreover, the LSI is an expensive eager algorithm of complexity $O((m+n)^3)$ which would need to be calculated regularly to take into account new material and users in the database.

Clustering

Several researchers have proposed pre-query clustering as a suitable off-line process for ACF (Kohrs & Merialdo 1999, O’Connor & Herlocker 1999, Ungar & Foster 1998a, Clerkin, Hayes & Cunningham 2001). The objective of a clustering task is the unsupervised partition of the dataset into a number of classes, none of which are specified beforehand. In a clustering scenario objects are typically represented as vectors of n feature values, and the similarity of two objects is defined as the Euclidean distance between them in n -dimensional space, although other similarity measures could be used. The goal of the clustering algorithm is to maximise inter-cluster distance while minimising intra-cluster distance. An example of such an approach is the k -means clustering algorithm (Jain & Dubes 1988). The steps of the algorithm are given below:

1. Select k objects, each of which initially represents a cluster mean.
2. Assign each remaining object to the cluster to which it is most similar, based on the mean value of the objects in the cluster (the cluster centroid).
3. The cluster centroids are then updated.
4. Repeat steps 2 and 3 until the same points are assigned to each cluster in consecutive rounds. The clusters can now be considered stable.

Using ACF data it is possible to cluster either users or items. If we view Table 4.4 on a row-by-row basis, users are represented as a vector of the items they have rated and can be clustered

accordingly. Likewise, if we view Table 4.4 on a column-by-column basis, each item can be represented as a vector of the scores assigned by each user.

Clerkin et al. use the k -means algorithm to build reduced user-profiles by firstly clustering items. Users are then marked up according to the amount of items they have in each of the clusters. This is a kind of dimensionality reduction technique that would speed similarity calculation significantly. Some promising results have been obtained using these reduced profiles on Smart Radio data (Clerkin et al. 2003). The key motivation of this work was in solving the new user bootstrap problem (Clerkin, Hayes & Cunningham 2001). In Chapter 6, we describe an alternative technique for addressing the same problem in which we reduce the item dimensions of the ACF matrix using a simple transform.

The Expectation Maximisation (EM) algorithm has been used as the basis for many unsupervised learning algorithms where there are unobserved values (Dempster et al. 1977, Cheeseman et al. 1988). Breese et al. use it on ACF data to cluster users (Breese et al. 1998). The assumption they employ is that within the dataset there is a mixture of k distinct normal distributions which represent groups of users that capture a shared set of preferences. The learning task involves finding a hypothesis that describes the means of each of the k distributions. Once the mean for each distribution is found, the maximum likelihood hypothesis for each distribution can be established. The problem involves deciding which instances were generated by which distribution.

Mobasher et al. (2000b) cluster users based on past web page usage, and then produce an *aggregated user profile* based on the centroid of each cluster. Each aggregated profile represents a cluster as a weighted collection of page views. The collaborative filtering based on the aggregated profiles results in improved query-time performance, but a reduction in predictive accuracy.

4.6.4 Lazy Similarity Measures

Several lazy similarity measures have been used in ACF to find neighbours. We will describe the three most commonly used.

Least-squares measure

Shardanand and Maes (1995) proposed a simple similarity metric that was based on the least-squares measure.

Equation 4.2

$$\text{sim}(U_a, U_u) = 1 - \frac{1}{|m|} \sum_{i=1}^m (U_{a,i} - U_{u,i})^2$$

where $U_{a,i}$ and $U_{u,i}$ represent the ratings for user U_a , and user U_u for item i , and where m is the number of items they have in common. This could be the basis for clustering users in an eager version of ACF or it could be used at run-time to identify neighbours within a threshold.

Cosine measure

Information retrieval systems require a measurement of similarity between documents. Salton et al. defined the similarity between documents in terms of the normalised inner product of two vectors, where each vector represents the keywords extracted from the document (Salton et al. 1975).

Equation 4.3

$$U \cdot J = |U| |J| \cos \theta$$

The smaller the angle, θ , between the vectors, the more similar they are. Accordingly, an angle of zero degrees will produce a cosine value of 1, maximum similarity, while an angle of 180 degrees will produce a cosine value of -1 which indicates maximum dissimilarity.

Equation 4.4

$$\text{sim}(U_a, U_u) = \cos \theta = \frac{U \cdot J}{|U| |J|} = \frac{\sum_{i=1}^m U_{a,i} \times U_{u,i}}{\sqrt{\sum_{i=1}^m (U_{a,i})^2 \times \sum_{i=1}^m (U_{u,i})^2}}$$

Typically, in information retrieval each term in the pair of vectors is weighted by the Term Frequency–Inverse Document Frequency (TF-IDF). This metric measures the relative frequency of occurrence of a word in the corpus of documents. When measuring document similarity, commonly occurring words are lightly weighted whereas words that occur less frequently receive greater weight. This captures the intuition that less frequently occurring words are a more useful indication of a subject. The cosine measure combined with TF-IDF has been used in automated collaborative filtering to measure similarity between users in which the components of the vectors contain scores for items rated by both users (Breese et al. 1998).

Pearson- r correlation

The Pearson correlation coefficient is a widely used means of grouping users in an ACF context. Pearson's correlation reflects the degree of linear relationship between two variables on a scale of -1 to 1 . The coefficient equation can be derived from Equation 4.2 where the components for each vector are deviations from the vote average of each vector, e.g. $U_{u,i} = (u_{u,i} - \bar{u}_u)$.

Equation 4.5

$$\text{sim}(U_a, U_u) = \frac{\sum_{i=1}^m (U_{a,i} - \bar{U}_a) \times (U_{u,i} - \bar{U}_u)}{\sqrt{\sum_{i=1}^m (U_{a,i} - \bar{U}_a)^2 \times \sum_{i=1}^m (U_{u,i} - \bar{U}_u)^2}}$$

Clearly, the Pearson coefficient as a measure of similarity is only appropriate for scalar vectors because of its reliance on average deviation, which is not appropriate in the context of the Boolean vectors illustrated by the rows in Table 4.5. The Pearson-*r* correlation is a measure of correlation rather than similarity. For that reason it can identify the correlation between user vectors with different means. The two example profiles illustrated in Table 4.6 each have rated four items on a scale of 0–1. Even though they have very different means, they have a high correlation score.

Table 4.6: Highly correlated users with different means

	user A	user B
item 1	0.1	0.4
item 2	0.6	1.0
item 3	0.3	0.7
item 4	0.2	0.4
Mean	0.3	0.625
Correlation	0.966988	

Users will typically have individual rating scales. Some users are more cautious about giving high ratings. The Pearson-*r* correlation allows us to identify users with similar ratings patterns with respect to their mean. In the case of binary data, the cosine measure is used (Breese et al. 1998).

4.6.5 Similarity Issues

Degree of overlap

Generally, the amount of items that any user will rate in an ACF domain will be small. The algorithms discussed work under the assumption that overlap occurs between profiles. In fact the overlap may be very small. A user may be similar to his/her neighbours based only on a portion of their profile, in which case neighbour users may not be similar at all. Figure 4.5 illustrates the problem using a very simple scenario. The intersection of the Venn diagram represents items that the users have used in common.

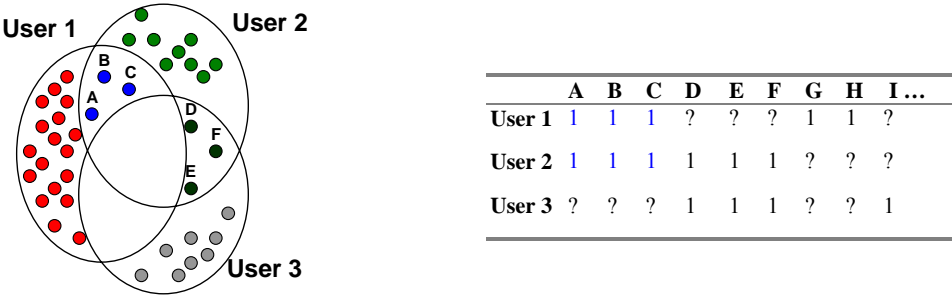


Figure 4.5: The Venn diagram illustrates the problem of measuring similarity based on few overlapping ratings

There are three users in the system. All have used a portion of the system assets. Yet they overlap only on a few items. When measuring the similarity between users we can only take the items on which they overlap into account. User 2 is deemed to be a neighbour of User 1 and User 3, based on an overlap of only three items each. These few items in question may not be representative of

each user's core interests, particularly so in this case where we are dealing with implicitly derived, binary data. Consequently, the recommendations made to User 1 from User 2's profile (and vice versa) may not be suitable.

Sparseness and transitivity

Figure 4.5 illustrates another problem with basing similarity measures on items in common: lack of transitivity. In the example, even though User 1 is similar to User 2 and User 2 is similar to User 3, we cannot infer similarity between User 1 and User 3 because there is no overlap between profiles. This is particularly problematic where the sparseness of rating data means that overlap is generally low. Some work has been done on an eager, single link clustering technique to overcome the transitivity problem in very sparse domains (Rafter et al. 1999). Using this technique each cluster is a maximal set of users such that each user has a similarity above a threshold with at least one other user in the cluster.

Establishing thresholds

Similarity threshold

Several research papers report improved predictive accuracy when thresholds are applied to the selection of neighbours (Balabanovic & Shoham 1997, Gokhale & Claypool 1999, Herlocker et al. 1999, Sarwar et al. 2000). A threshold can be used either to select neighbours above a particular similarity score or to select k -nearest neighbours. Setting the threshold appears to be domain dependent and is generally a trial and error procedure which requires off-line evaluation of metrics such as mean average error and precision/recall (Herlocker et al. 1999, Sarwar et al. 2000b).

Overlap threshold

Where data is relatively plentiful, setting a minimum overlap threshold that prospective neighbours must meet has been shown to improve predictive accuracy (Balabanovic 1997, Gokhale & Claypool 1999). The key idea is that a neighbour with more items in common is likely to be a better predictor. The process of deciding the threshold involves evaluating the predictive accuracy and the number of items predicted at different threshold levels. Setting this threshold too high when data is very sparse may lead to no neighbours being selected for certain users.

Significance weight

Herlocker et al. (1999) introduce a *significance threshold* to take into account the fact that correlation scores between two users may be based on very few items in common. If the number of co-rated items, m , is below the significance threshold, t , the correlation between the users is devalued by a *significance weight*, w , given in Equation 4.6

Equation 4.6

$$w = \frac{m}{t}$$

If the number of items rated in common is above the threshold, the significance weight is set to a value of 1, ensuring that the correlation score is preserved. The significance threshold must be established by trial and error for each dataset.

4.7 Extraction of Candidate Items

Since the amount of candidate items that could be retrieved from neighbour profiles may be large, some heuristics may be applied such as retrieving the n -most frequently rated items in the neighbourhood set, or by checking the overall correlation score for those neighbours that have endorsed a data item. Another rule of thumb in extracting candidate items would be to set a date limit on items under consideration and only extract data items recently endorsed by other users. If content representation is available, only candidate items of a certain type (e.g. genre = jazz) could be extracted. Again with this technique there is the risk of retrieving no items at all, or retrieving items that will fare badly in the next step of the ACF process.

4.8 Producing Predictions

Once a neighbourhood set has been selected, it can then be used to produce ratings for a series of assets. The predicted value for a rating on item i by the active user, U_a is given by Equation 4.7

Equation 4.7

$$U_{a,i} = \bar{U}_a + \frac{\sum_{u=1}^t (U_{u,i} - \bar{U}_u) \sigma_{a,u}}{\sum_{u=1}^t |\sigma_{a,u}|}$$

where t is the number of neighbours that have rated item i , an item not previously rated by U_a . The equation is a weighted aggregation of the ratings expressed by the neighbourhood set for this item. Each neighbour's ratings is normalized by subtracting their average rating, \bar{U} and weighted using their similarity to U_a , $\sigma_{a,u}$ (Resnick et al. 1994).

4.8.1 Top N Predictions

The top N items are sorted according to the prediction score for each candidate item. However, a measure of confidence needs to be considered for each prediction, since a prediction may have been calculated using a low number of neighbour ratings. A simple confidence measure would indicate the fraction of the neighbourhood set which contributed to the prediction. This confidence level could be made available to the user receiving a recommendation based on the prediction.

Herlocker et al. have shown that providing recommendation explanations improves user confidence in the ACF system (Herlocker et al. 2000).

4.8.2 Most Frequent Item Recommendation

Sarwar et al. suggest a very simple technique to find the top N items based on a neighbourhood of users. The recommender system returns the n most frequently (positively) rated items in the neighbourhood set that have not been rated by the user (Sarwar et al. 2000).

4.9 ACF issues

4.9.1 Recommendation Latency

In Section 4.3 we demonstrated how collaborative filters differ from content-based filters. While not using explicit content descriptors, collaborative filters tap the preferences displayed by similarly minded users in the system. However, in order to keep the filter from running dry, users must have an alternative means of receiving new items of interest as well. Intuitively, for items to be recommended to one user in an ACF system, other users must have already rated them. New items in the system will never be recommended using ACF unless they have been ‘discovered’ by other users. Furthermore, there is little incentive for the user to provide early ratings on a new item since the information will not improve his/her recommendations (Avery & Zeckhauser 1997). This is called the latency or early rater problem in ACF. To counter this, a secondary means of drawing the active user’s attention to new items of interest and encouraging feedback needs to be considered. The simplest strategy would be to occasionally include random items, or new items in a batch of recommendations. From an ACF perspective, receiving negative feedback on such items can be as valuable as receiving positive feedback since it ultimately affects the degree of correlation between neighbour users. However, from the user’s point of view such a strategy may destroy confidence in the system’s recommendation ability unless flagged as a ‘profile elaboration’ exercise. As we will discuss in Section 4.10, using a parallel content filter or search facility can alleviate this problem.

4.9.2 Bootstrapping the System

There are no standard techniques for starting an ACF system from scratch. Typically, the ACF recommender works in conjunction with another means of finding system assets such as a search engine, or a content-based recommender. Once sufficient data has been collected as a result of users availing of these parallel techniques, the ACF recommender can be engaged. However, ACF requires a certain amount of data before it will work well. There are two aspects to the quantity of data needed for ACF systems: the number of users, and the sparsity of the data. The former influences the neighbourhood selection process – if there are few users in the system the ability to assemble a well correlated set of neighbours for *each* user is diminished. Even with a lot of users, if

there is not much rating data, the ACF algorithm is constrained by the *sparsity* of the dataset. In the extreme case, this means that there is insufficient data to calculate correlations between users. Since data sparsity is a given with ACF systems, the issue becomes one of degree.

4.9.3 The New User

Even if there is other user-data in the system, a new user cannot receive recommendations until he/she is correlated with other users. Bootstrapping new users into the system is important, in that many users may not have the patience to build up a profile whilst receiving poor recommendations. Several solutions have been proposed. Shardanand & Maes (1995) offered the new user an equal mix of the most popularly rated artists and random artists taken from the artist database, and encouraged the user to rate these to receive a personalised selection. A preferable approach would be to offer items that appear to be key discriminators of user preference within the system, or with a cluster of users. Inductive techniques based on decision trees (Quinlan 1993) are generally successful in determining the discriminating features in datasets. However, such analyses do not operate well where there are many missing values. Clerkin et al. propose using cluster data to bootstrap new users (Clerkin et al. 2001b). By eagerly clustering users and then extracting the n most frequent items in each cluster the ACF system can offer a new user a selection of popular items rated by users with differing tastes.

4.9.4 The Grey Sheep

Clearly collaborative recommenders will provide good recommendations for users who are well correlated with a group of other users. However, Claypool et al. identified the concept of the grey sheep, a user who does not consistently agree or disagree with any group of users in the system, and who rarely gets accurate collaborative recommendations (Claypool et al. 1999). Intuitively, this user lies on the cusp of existing clusters of users within the data. This phenomenon could be argued to be the result of insufficient user data, and that all new users are grey until they find sufficiently close neighbours. However, in the next section we discuss the hybrid of content-based techniques and collaborative techniques which have been shown to ease the problems caused by latency, new users and grey sheep.

4.10 Combining Collaborative and Content-Based Techniques

In the previous chapter we discussed CBR, an AI technique which can be used as a content-based recommendation strategy. One of the strengths of a content-based strategy is that it will consistently find items that match a given user profile. Thus, new items can be recommended to the user, irrespective of whether other users have endorsed them. This addresses the latency problem in ACF systems whereby new items cannot be recommended until they have been rated by a number of users. Content-based systems have the drawback that they will only tend to recommend the types of items described in the user profile, thus limiting the user to the same type of items he/she has

used in the past. ACF, on the other hand, taps into the collective experience of a neighbourhood of users and can make much more diverse recommendations. Of course, it could be argued that a technique like CBR could make diverse recommendations if a sufficiently elaborate domain ontology were developed. However, the knowledge engineering required to model the relations between items in a changing domain, as well as the utility measures to reflect each user's preferences would be enormous. Instead, ACF captures this knowledge implicitly.

Several research applications have demonstrated that the weaknesses of content-based systems and of collaborative systems can be alleviated by the strengths of each respective technique (Balabanovic & Shoham 1997, Smyth & Cotter 1999a,b, Pazzani 1999). These strengths and weaknesses have been summarised in Table 4.7. to illustrate the mutually beneficial affect of each technique.

Table 4.7: The strengths (+) and weaknesses (-) of collaborative filters and content-based filters

Content-based (CB)	← comment →	Collaborative Filter (CF)
+ Consistently finds items similar in type to those liked in the past. - Cannot infer whether the items are good or bad quality.	Content filters can recommend new items irrespective of their rating history. However, the content representation cannot capture subjective measures such as aesthetic quality.	- Suffers from new item latency - 'Grey sheep' problem + Recommends items using endorsements from like-minded users.
- Cannot infer user's preference for new types of content.	ACF can suggest cross 'genre' recommendations which a content filter would never find.	+ Finds content type other than that used by the user in the past.
- Suffers from new user bootstrap problem, but can start retrieving 'similar' content quickly. + Independent of the ratings of other users in the system.	Both systems require historical data to begin recommending/filtering content. However, content-based techniques can 'query by instance', i.e. find new items 'similar' to a single instance chosen by the user. ACF systems require rating data from the active user and other like-minded users	- New user bootstrap problem - Requires large amount of rating data from other users.

4.10.1 Combination Methods

Burke has surveyed the various techniques for combining ACF with other recommendation strategies (Burke 2002). He defines 6 combination techniques described in the literature. We will outline these techniques, as they are relevant to our discussion in the next chapter of the hybrid approach used in Smart Radio.

Weighted:

With this technique a prediction on an item is made using a weighted average of the predictions of each of the recommender techniques in the system. The *P-Tango* system, a personalised newspaper system, learns the appropriate weights for the ACF filter and the content filter by periodically evaluating the success of each filter on the use-data collected by the system for each user. Thus each user is assigned different weights for the ACF filter and content filter according to the success of each technique operating on its own (Claypool et al. 1999). Claypool suggests that this

technique is appropriate for dealing with grey sheep in which case the content filter will predominate until the ACF filter has accumulated enough data to make good predictions.

Switched:

Switched systems use a measure of quality to provide recommendations from one recommender technique or the other. The *DailyLearner* system uses a content-based strategy as the primary recommendation strategy. If the recommendation cannot be made with enough confidence, it falls back on a collaborative strategy (Billsus & Pazzani 1999).

Mixed:

Mixed combinations occur where several recommendations are presented together. For instance, the *PTV* system employs a mixed strategy to compile a personalised TV guide for each of its users. PTV arbitrates between possible conflicts by allowing the content-based strategy take precedence over the collaborative strategy (Burke 2002).

Feature combination:

Feature combination systems are essentially content-based techniques in which the collaborative data is treated as an additional set of features. Basu et al. (1998) apply an inductive rule learner to such a merged dataset and report significant improvements in precision over purely collaborative techniques. However, in order to achieve this, content features were manually selected.

Cascade:

Cascading combinations employ one recommendation strategy to retrieve a candidate set of recommendations that is then refined by a second recommendation strategy. Burke's revised Entrée system, a restaurant recommender, employs a cascaded combination. The content-based recommender produces an ordered list of recommendations, while the collaborative recommender is used to decide the ordering of tied recommendations (Burke 2000).

Feature augmentation:

Feature augmentation refers to a process where the output of one recommendation strategy contributes additional features to the input of a second recommender. The GroupLens research group has employed content-based filter agents specialised in particular topics as pseudo-users to rate new content. These ratings alleviate the sparsity problem and the latency problem for the collaborative filter recommender (Sarwar et al. 1998).

Meta-level:

In this case the model generated by one recommender is used as the input for another. An example of this technique is the Fab system described earlier (Balabanovic & Shoham 1997).

Finally, it should be noted that building a hybrid model supposes that good content descriptors are readily available which might not be the case for certain domains. The quality of either type of data (content or collaborative) will be the primary factor in deciding a combination strategy.

4.11 Data Ordering and Context Insensitivity

One serious drawback of ACF is that it is not sensitive to a user's context. We define context as the discourse that informs the user's current behaviour in the system – their immediate requirements, their motivation, their previous experience, their preferences and the knowledge available to them (Hayes et al. 2002b). 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. In fact 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 the user's accrued interests. This may not be a real drawback if we are using ACF in its role as a passive filter. However, where ACF is required to produce recommendations, its lack of sensitivity to the user's current interests may cause frustration and distrust. In Chapter 5 we will discuss this problem further and elaborate our solution which involves employing a lightweight CBR strategy to order the ACF recommendations.

4.12 Observations on ACF and CBR

It could be argued that the representation issue is not such a defining difference between ACF and CBR given that some k -Nearest Neighbour implementations determine similarity, as ACF does, with no reference to the semantics of the case features. The difference in representation may be only one of degree. Each ACF user profile represents a user's 'consumption' history to date. The goal of ACF is to recommend the next step the user should take based on the items in his/her profile up to this point. This temporal perspective on ACF has generally been neglected. As such we suggest that ACF could be viewed as an attempt to model usage patterns where the goal is to suggest the next step in an ongoing process of use. From a CBR perspective we could view the ACF recommendation process as a *case completion* process – an incremental elaboration of the user profile based on feedback given by the user (Burkhard 1998). In Chapter 6 we will pick up this topic again when we describe the data architecture of our ACF implementation.

4.13 ACF Systems

4.13.1 GroupLens

A collaborative filtering system for larger communities cannot rely upon people knowing each other. Researchers in the University of Minnesota published the first research on a Collaborative Filtering System that automated the process of finding users with similar interests (Resnick et al.

1994). The GroupLens system was in fact an open architecture for the filtering of Usenet articles. It consisted of a client news reader that allowed users to rate articles and a server side component called a *Better Bit Bureau* that made predictions for the participating user on each incoming Usenet article from a news server. The Better Bit Bureau made predictions using a weighted aggregation of the opinions of neighbour users on the article. The client reader sorted the Usenet threads according to the maximum predicted score over the articles in the thread. Since users typically spend very little time on any single article, a simple keystroke rating mechanism was supported by the client reader. To counter the data sparsity problem, later versions of the GroupLens system experimented with gathering implicit ratings based on the length of time a user spent reading an article (Konstan et al. 1997) and employed content-based filter agents as pseudo-users to rate articles (Sarwar et al. 1998).

4.13.2 Ringo

Ringo was a text-based music recommender system developed at the MIT media lab (Shardanand & Maes 1995). Although it was developed independently from the GroupLens project, both systems appeared almost at the same time. Shardanand and Maes described their ACF techniques as algorithms for automating “Word of mouth” recommendations. Although they evaluated several similarity-based algorithms, their approach was much the same as the GroupLens project. Ringo allowed its users to rate artists by email and receive messages containing recommendations for artists and albums in turn. The web-based incarnation of Ringo formed the basis of the Firefly network (www.firefly.com), a popular community-based portal where on-line-users could receive recommendations on music, books and movies. The Firefly network was acquired by the Microsoft Corporation in April 1998, and subsequently closed. Its popularity is attested to by the fact that there is a ‘museum’ website for former Firefly members³⁶.

4.13.3 Bellcore Video Recommender

The researchers behind Bellcore Video Recommender took a Computer Human Interaction (CHI) perspective on the collaborative filtering task, viewing it as a means of transferring information within a virtual community (Hill et al. 1995). They defined several collaborative recommendation implementation and design goals which would enable people to share some of the informational benefits of belonging to a community without having to accrue the associated communication costs, such as time taken to establish a personal relationship. The key ideas behind the Bellcore system were that recommendations should not be anonymous. The recommender system should make it clear that recommendations originate from people, rather than a black box and there should be a clear indication of how much confidence to place in them. This explanatory capacity of recommender systems is once again receiving attention (Herlocker et al. 2000). A novel aspect of

³⁶ <http://www.myfireflytown.com/>

the Bellcore system was that recommendations could be elicited to fit the preferences of a set of people rather than just an individual. Hence the system could be queried for a video that would best suit the collective preferences of a group of friends. The Bellcore project also introduced to ACF research the idea that there is an upper bound to the predictive accuracy of any system that operates with user ratings given that all ratings contain noise, i.e. people may give different ratings a week later for the same items.

4.13.4 Fab

Fab was an agent-based information retrieval system for the delivery of personalised web pages developed at Stanford University (Balabanovic & Shoham 1997). It used a hybrid of content-based retrieval and ACF to retrieve a customised set of web pages for each of its users. Primarily, the Fab system was a content-based information retrieval system that used a set of collection agents to actively gather content on specialised topics. These agents would retrieve topic-specific pages, represent them using the vector space model (Salton et al. 1975) and then distribute them to users whose profiles sufficiently matched. The standard IR cosine measure (see Equation 4.4) was used as a similarity measure between pages and profiles (which were also represented using the vector space model). Using the same cosine measure, similarity between the content-based user profiles was calculated in order to establish a set of nearest neighbours for each user. When users were presented with their selections they were required to rate them. This feedback allowed their user profiles to be refined. Pages receiving a high rating were directly sent to a user's nearest neighbours. A selection agent on behalf of each user ensured that duplicate pages, multiple pages from the same web site, and previously viewed pages were removed before a set of recommended pages was presented to the user. By using the collaborative approach, Fab was able to leverage the knowledge of its users to make recommendations that would have escaped the content-based recommendation strategy.

The Fab system illustrated how the information filtering task could be viewed as a process of long-term information retrieval (Belkin & Croft 1992).

4.14 Conclusion

This chapter has illustrated the techniques and issues surrounding automated collaborative filtering as an on-line strategy for filtering or recommending content. ACF differs from content-based filters, such as CBR, in that it does not rely upon representations of the items to make recommendations. Instead it uses the ratings of other, similarly minded users to predict whether a user will like an item or not. Ratings can be gathered explicitly or implicitly. While explicit information may contain more reliable information, it forces the user to work, which he/she may not be willing to do unless they are sure of the benefits. Two types of ACF were discussed: model-based and memory-based. Memory-based collaborative filtering has traditionally performed well in terms of prediction, and ability to adapt quickly to user feedback. However, as the data in the

system grows, like most lazy learning systems, they impose unacceptable query-time delays. Therefore research has been carried out on implementing partially lazy, and full model-based ACF algorithms. Model-based algorithms have short query time responses but their training times may be too long to deal with a system in which new data is continuously arriving, and fast learning is required. While some eager algorithms out-perform memory-based techniques in terms of predictive accuracy in off-line situations, their efficacy in a real time system has yet to be demonstrated. We introduce the idea, which we develop in a later chapter, that the ACF methodology is similar to an interactive, incremental CBR technique called case completion.

ACF techniques have some serious drawbacks. They require a lot of data to work since the user-item matrix is typically very sparse. They cannot recommend items that have not been rated by another user in the system, nor can they produce recommendations for a user until the user has built up a ratings set. If a user cannot be correlated with a group of users, recommendations will generally be poor. These weaknesses can be alleviated by the use of a content filter working in parallel, if content description is available in the first place.

We describe how ACF is insensitive to user context and will make recommendations based on the entire user history. These recommendations may be inappropriate to what the user is interested in at the moment. In the next chapter we will describe the concept of user *context* in more detail, and show how we augment the ACF technique using CBR to provide context-sensitive recommendations.

Chapter 5: Recommendation in Smart Radio

Acting upon recommendations from other people is a normal part of life. We do it when we eat at a restaurant on the advice of a friend, or we see a movie having read the review in the newspaper of our choice. In each case our decision to act upon a recommendation is based on a matter of trust:

1. We trust that the recommender has sufficient knowledge of our tastes or of the tastes of people like us;
2. We trust that the recommender has knowledge of the alternatives available.
3. We trust that the recommender is acting consistently and with our interests in mind.

By using recommendations we can take a shortcut to the things we like without having to try many things we dislike or without having to acquire all the knowledge to make an informed decision. Irrespective of the techniques used, the success of the *automated* recommender is still reliant upon the trust of the user, having sufficient knowledge of the user's requirements, and having knowledge of the range of items available.

This chapter will describe the recommendation strategies employed in the Smart Radio recommender system. In Chapter 3 and Chapter 4 we examined the pros and cons of content-based and 'content-less' recommendation services. Smart Radio makes use of both strategies to recommend selections of music to its listeners, and to allow its listeners to find new music. We introduce the unit of recommendation in Smart Radio, the playlist, a compilation of 10 music items designed to give about 40 minutes of listening time. Unlike most examples of recommender systems, the Smart Radio System is concerned with recommending composite objects rather than single objects.

Given that recommender systems generally operate as augmentative systems within larger application systems, their purpose is to help the user exploit the resources available within the larger application domain. Therefore, when employing a recommender system, the goal is to translate continued user satisfaction into continued use of the system resources. Typically recommender systems are designed to recommend new items only. However, in the case of Smart Radio this strategy is inappropriate for listeners who prefer a music mixture that includes items they have previously listened to. We describe three techniques for managing a domain where previously 'consumed' items may be recommended again (*novelty weight*, a *re-use window*, *refractory periods*).

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 Automated Collaborative Filtering (ACF) where the similarity between users is leveraged in order to make recommendations. Apart from the

obvious advantage of a ‘knowledge-light’ approach to recommendation, ACF is often attributed as being able to 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 are difficult to encode into content-based systems. As we have described in Chapter 4, ACF does have well documented drawbacks such as the problem of bootstrapping new users and new content into the system. Furthermore, ACF techniques are not sensitive to the user’s ‘local’ or contextual requirements. One of our primary concerns was to leverage the light content description available to us to improve this aspect of the ACF recommendation process. Accordingly we describe our implementation of a novel hybrid recommendation system in which a lightweight case-based strategy provides context-sensitive ordering to the recommendations provided by the ACF module.

5.1 The Playlist

The unit of recommendation in Smart Radio is the *playlist*, a compilation of 10 music tracks assembled by one listener and recommended to other like-minded listeners.

The design goal in Smart Radio was to provide a personalised service of streaming music using a recommendation system to suggest suitable compilations of music to listeners. Smart Radio uses a streaming service which has the advantage in that it allows a *programme* of music to be immediately delivered to the listener. Rather than automatically compiling playlists, the Smart Radio approach is to allow listeners to compile playlists that can then be recommended to other listeners using a combination of ‘content-less’ and content-based recommendation strategies. By using a playlist of music as the unit of recommendation, the work and knowledge involved in putting together a new compilation of music is distributed to other listeners. Our original hypothesis was that the playlist format 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’. As we discussed in Chapter 4, this type of aesthetic knowledge is difficult to formalise. However, this hypothesis has been tested to breaking point by the eclectic tastes of our listeners.

The playlist format is also attractive in that it also allows individual users to become *producers*, compiling selections of music with the view that the selection will be passed on to other listeners. This is often reflected in the titles Smart Radio listeners give to their playlists (Figure 5.1). As we discuss in the next chapter, users who actively engage with the community in this way are central to keeping the recommendation process running in Smart Radio.

In terms of music delivery the playlist is a well understood format. Music is generally played as part of a collection, whether on a CD or on the radio. Popular music compilations repeatedly rate highly amongst the best selling CDs every year. Despite this, the music industry has never put in place techniques whereby people could purchase custom-made compilations, even though this activity was commonly carried out by consumers using blank tapes and, of late,

recordable CDs. A common offshoot of this activity is that homemade compilations are swapped or given as gifts despite the fact that this contravenes copyright. Furthermore, the increasing use of peer-to-peer file-sharing software has enabled the web-savvy music listener to pick and choose music tracks at will. However, unlike file-sharing networks, a streaming service like Smart Radio means that delivery and playback happen at the same time.



Figure 5.1: A playlist built by Smart Radio user, *coyle*, dedicated to another user, *Patrick*

5.2 Recommending Playlists

5.2.1 Simple ACF Strategy

Each listener in Smart Radio has a listening history consisting of a number of playlists made up of 10 component tracks. The listener may have explicitly rated individual tracks in each playlist, or have played tracks repeatedly from which we can implicitly infer feedback as we describe in Section 6.4. Using this data we can build a user–item matrix similar to that illustrated by Table 4.4. However, since playlists are composite objects, correlations are made between listeners based on the artists or tracks in their playlist history rather than on the playlists themselves. Two listeners may be highly correlated based on the overall component tracks/artists they have listened to, even though they may not have listened to the same playlists.

Correlation base

In Chapter 4 we discussed the problem of data sparsity in ACF systems. There are two principle effects of data sparsity. One means that correlations between users may be based on very few items in common. Secondly, basing correlations on single items does not take into account the latent relationship between items. Billsus and Pazzani used Latent Semantic Indexing to reduce the

dimensionality of a ACF user-rating set, thus implicitly discovering the higher level relationship between items based on user ratings (Billsus and Pazzani 1999).

Since all our music tracks are indexed under artist and genre titles, by default we correlate users on the artists they have in common. The effect of this is to reduce the ‘item’ dimension of our user-rating matrix from 4131, the number of tracks in the database, to 333, the number of artists. As we see in Chapter 7, this does not cause a significant change in prediction accuracy of our ACF algorithm. It does mean much quicker neighbourhood computation, lower memory overhead and, importantly, a much faster bootstrap process for the new user. These issues will be discussed in greater detail in the next chapter, *System and Data Architecture*.

Making a prediction

Making a prediction on a playlist involves making a prediction for each of the component tracks where we do not have any explicit or implicit feedback. To make a prediction for each track we can employ Equation 4.7, where the vote is the weighted average of the normalised vote of each neighbour.

Table 5.1 illustrates a typical playlist where there are three tracks that the user has rated previously and where we have made predictions for six 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 listener’s average rating.

Table 5.1: A playlist with known and predicted scores

Track Id	Status k/p	Score
127	k	0.8
23	p	0.4
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	0.6

As shown in Equation 5.1, the overall prediction, op , for the playlist is the sum of the predicted and known scores for the tracks in the playlist, divided by the number of tracks in the list.

Equation 5.1

$$op = \frac{(\sum k + \sum p + \sum u)}{n}$$

Thus, the overall score for the playlist in Table 5.1 is 0.7. However, the playlist recommendation set is *ranked*, not according to the overall score of the playlist but according to a *weighted score* where the weighting refers to a user-defined constraint that we call the *novelty weight*.

Novelty weight

This weight was introduced in response to listener criticism that an earlier version of Smart Radio did not take into account listener preferences for receiving new music. Unlike other recommendation domains the recommendation service in Smart Radio is firmly tied to its simultaneous content delivery service. People receive recommendations and can act upon them immediately. Given that people normally listen to music they like more than once, the recommendation strategy employed should adapt to take this into account. Typically recommender systems are designed to recommend new items only. However, a strategy biased towards recommending new items only may be disagreeable to users who prefer a music mixture that includes items they have previously listened to. Given that a playlist is a composite object, it will typically contain a mixture of new music items and items the user is familiar with. When making an overall prediction for the playlist, a decision needs to be made on how to weight the known scores with respect to the predicted scores of the composite tracks. In Smart Radio v1 (Hayes & Cunningham 2000), we viewed the task as purely that of recommendation of new music. Thus we gave greater weight to predicted items in the playlist which produced recommendation sets biased towards playlists made up of a lot of ‘unknown’ music. For some listeners this was a benefit, but for others receiving such recommendations required an investment that they were unwilling to make.

To address this issue, the Smart Radio system now allows listeners to indicate a preference for the amount of new music (*novelty*) they wish to receive in each playlist. The novelty weight lies on a scale of 0 to 1 where a higher score indicates a partiality for new music in each playlist.

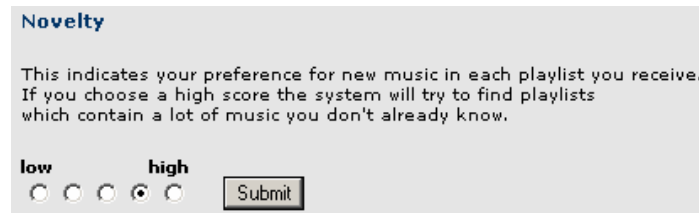


Figure 5.2: The novelty preference in Smart Radio

Equation 5.2 gives the weighted prediction, w_p , for the playlist in Table 5.1 where w refers to the *novelty weighting*.

Equation 5.2

$$w_p = \frac{(1-w) \sum k + w (\sum p + \sum u)}{n}$$

Using Equation 5.2, the score for the playlist where the listener has specified a low novelty factor (0.25) is 0.285.

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

The score for the playlist where there is a relatively high novelty factor of 0.75 is 0.415.

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

The playlist recommendation sets are then ranked according to the *weighted prediction* of each playlist, and the top N playlists can be displayed to the user. Thus the recommendation strategy is guided by the user's preferences. Swearingen and Sinha's study of user-satisfaction in recommender systems (Swearingen & Sinha 2001) demonstrated that users were inclined to be more accepting of automated recommendation where 'trust-generating' items were included in the recommendation set. In this context we can view the *novelty weight* as a technique for including 'trust-generating' items.

The realisation that the recommendation task should be sensitive to how people use the system assets, led us to redevelop the Smart Radio recommendation strategy to cater specifically for our listeners. In the next section we describe four further techniques to this end: *context sensitivity*, *playlist-reuse*, *refractory periods* and *adaptation flagging*.

Playlist reuse window

Given that people normally listen to music they like more than once, Smart Radio is able to recycle playlists that people have played in the past. This is different from most recommender systems or information filtering systems where it is expected that the recommendation set will always contain previously unseen items. Smart Radio has the facility to recycle playlists every 30 days. On each recommendation request the recommendation algorithm checks whether two conditions are met.

- 1) That the top rated ACF playlist has an un-weighted prediction greater than the average track rating calculated for the user.
- 2) That the number of recommendations returned is greater than 2/3 of what was requested. Each ACF request looks for 75 playlists. These are then sorted according to context, as we describe in the next section, and the top lists displayed to the user.

If either of these conditions evaluates to false, the system re-samples the candidate lists including lists the user has already listened to outside the previous 30 days. This is repeated until the conditions are met.

This technique caters for users who use the system heavily, exhausting the supply of new playlists that are being produced by their neighbours. As we discuss in Section 6.5.2, new playlists fuel the recommendation process, and their scarcity can be a bottleneck in recommending new material for users who use the system a great deal.

5.3 Context Boosting ACF

As we discussed in Chapter 4, 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 months or years. The basic ACF methodology involves lazily making recommendations using the full user history. However, ACF is not sensitive to the ordering of the usage data. The ACF algorithm assumes that the unordered dataset represents the *cumulative* interest at time t whereas only a portion of the data may be pertinent to the user’s requirements at time t^n .

Although the data informing an ACF profile does have an ordering relation (it represents a trace of the user’s usage pattern over a time period), this relation is not represented in the ACF profile. Correlation between users is based on the unordered set of items in each profile so that any information associated with the original ordering, including the user’s current interests, is discarded. Thus, ACF recommendations are produced without sensitivity to what items would be most useful to the user according to their most recent activity.

The diagram in Figure 5.3 illustrates this issue. The topmost coloured bar indicates the range of ‘topics’ in which a particular user is engaged over time in a generic domain. The coloured bars represent the *type* of the item in use at a point on the time line.

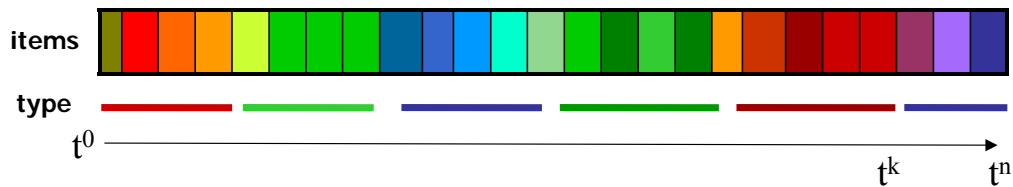


Figure 5.3: The ACF algorithm assumes that items of all types (coloured bars) are equally important at time t^n

This example is based on the presumption that we can determine *type* easily from content description of the items being used. The typical ACF-based recommender system may make recommendations for items of all types at the current query time t^n . However, in many domains where a user may be engaged in specialised activity, such as reading documents on a particular subject, or listening to music of a particular genre, many of these recommendations will be inappropriate and untimely. The problem is how to determine what the user’s interests are at a particular point, and make recommendations that cater to those specific interests. This 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.

One technique to address this might be to use only a relevant portion of the user’s profile (for instance items consumed between t^k and t^n) and then make ACF recommendations using this reduced profile. Isolating the subset of items that are relevant to the current listening context will be difficult, particularly in a domain where there is not much content description. Secondly, there is no guarantee that our user will receive recommendations of the relevant type if his nearest

neighbours also have eclectic interests. This technique has the added drawback that few neighbours may be correlated on a subset of the user-profile. An alternative technique would be to correlate neighbours based on a full user profile, and then extract candidate items of the relevant type. Again this requires content description to be available.

5.3.1 Context

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 in 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 in the field of Information Retrieval 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 Internet Explorer. Using information retrieval techniques for content analyses, *Watson* automatically issues requests to search engines in order to retrieve related material. Another such system is *Letizia* (Lieberman 1997, Lieberman 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 set of links that indicate a predicted preference ranking according to the current state of the profile.

Both *Watson* and *Letizia* have been termed ‘Reconnaissance’ 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 of recommendations 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 an off-line manner. In Chapter 7 we present an evaluation technique suited to measuring the success of contextually motivated recommendations.

5.3.2 Context in Smart Radio

As we illustrate in Figure 5.4, 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*. The key issue in this is to determine a representation for the user's current interests.

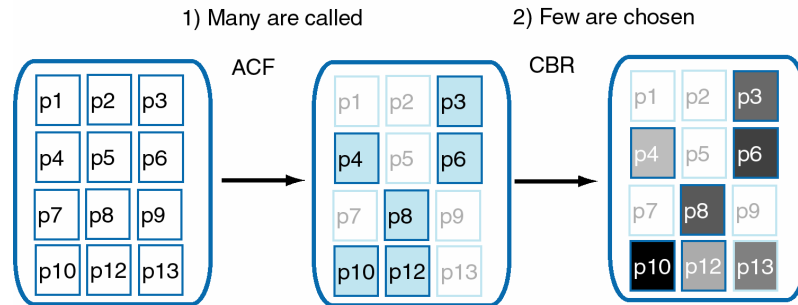


Figure 5.4: ACF primes a portion of the Case Base. Primed cases are ranked by similarity to the user's context.

Unlike the examples of the *reconnaissance* aides described earlier, which used automated content analysis of text documents to build a short-term user profile, the Smart Radio domain suffers from a deficit of good content description. Our objective is to enhance the ACF technique where very little content is freely available, and where the knowledge engineering overhead is kept to a minimum.

Task boundaries/Session boundaries

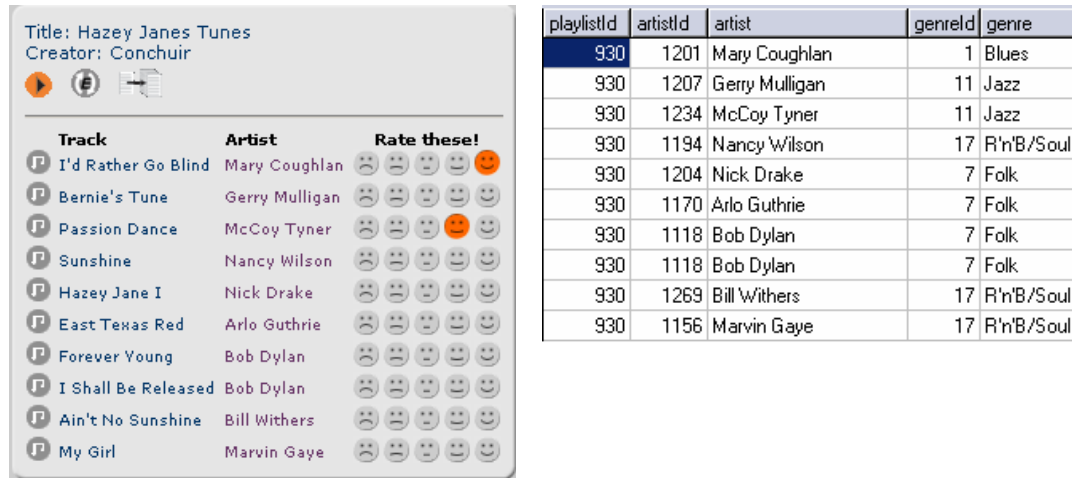
One of the difficulties in determining context is the delineation of where a user task ends and another begins. To illustrate this we will look at this in a situation where we have a large amount of content description. In information retrieval terms we can view each task as being composed of a series of subject-related queries. Were we able to delineate k primary areas of interest based on a long-term analysis of retrieved pages, we would still have the problem of determining when a retrieval task associated with one subject area begins, given that the earliest trigger would be in the form of query terms, which are often very sparse. Once the user has started to read retrieved pages, however, a context analyser module would have a much better indication of the subject area being researched since the page content will be much richer than the query terms.

In the case of the music items in Smart Radio, we make use of the meta-information 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. The type of information typically 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 needs to be scanned for inaccuracies in spelling and genre assignment. Furthermore, we do not use the potentially useful *date* feature since it is often missing or inaccurate.

Context event

Smart Radio is a closed domain with a finite number of playlist resources. By setting a playlist to play 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 *Letizia*, we transform the playlist representation into a *case-based* representation where the case features indicate the genre/artist mixture within the playlist. Given that 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.

By using the most recently played playlist as an indicator of the user's current interests we also solve the practical problem of having to develop a user-profile representation which is compatible with the representation used by the playlists.



playlistId	artistId	artist	genreId	genre
930	1201	Mary Coughlan	1	Blues
930	1207	Gerry Mulligan	11	Jazz
930	1234	McCoy Tyner	11	Jazz
930	1194	Nancy Wilson	17	R'n'B/Soul
930	1204	Nick Drake	7	Folk
930	1170	Arlo Guthrie	7	Folk
930	1118	Bob Dylan	7	Folk
930	1118	Bob Dylan	7	Folk
930	1269	Bill Withers	17	R'n'B/Soul
930	1156	Marvin Gaye	17	R'n'B/Soul

Figure 5.5: A playlist represented in terms of its genre/artist composition

5.3.3 Case Representation

We have two feature types associated with each track, `genre_` and `artist_`. The meaning 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. Such a case representation is shown in Table 5.2. 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. For instance the fact that

genre_7 dominates this playlist is not at all represented here. Thus our case description must contain the quantities of individual genres 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.

Table 5.2: A playlist representation with no information on the quantities of genre or artist types

Case Id	playlist_930
genre_	g_1, g_11, g_17, g_7
artist_	a_1201, a_1207, a_1234, a_1294.... a_1118

The case representation we used in Smart Radio is illustrated in Table 5.3. We have combined the features (genre_, artist_) and values from Table 5.2 to produce a representation that captures the composition of the playlist in terms of the quantity of genres and artists present. This representation also allows each case to be represented in a retrieval-efficient memory structure such as a Case Retrieval Net which we discuss further in Section 5.4.

The case mark-up demonstrated in Table 5.3 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). In this example only a portion of the artist_ features are shown. All cases in Smart Radio are represented in the CBML format. More recent work on this format is described in Coyle, Cunningham and Hayes (2002).

Table 5.3: CBML representation of a playlist with genre_/artist_ feature types

```

<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>
  <attribute name="artist_1207">1</attribute>
  <attribute name="artist_1234">1</attribute>
  ...
  <attribute name="artist_1118">2</attribute>
</attributes>
</casedef>
</case>

```

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. Figure 5.6 gives a simple example using just the `genre_` features. The target playlist containing 5 *folk* tracks, 3 *jazz* tracks and 2 *alternative rock* tracks is similar to playlist A with 4 *folk* tracks, 4 *jazz* tracks and 2 *alternative rock* tracks, but less similar to playlist B with 1 *folk* track, 3 *jazz* tracks and 6 *alternative rock* tracks. Although each candidate playlist contains the same genres as the target playlist, the proportion of genres in playlist B is clearly less close to the proportion of genres in the target.

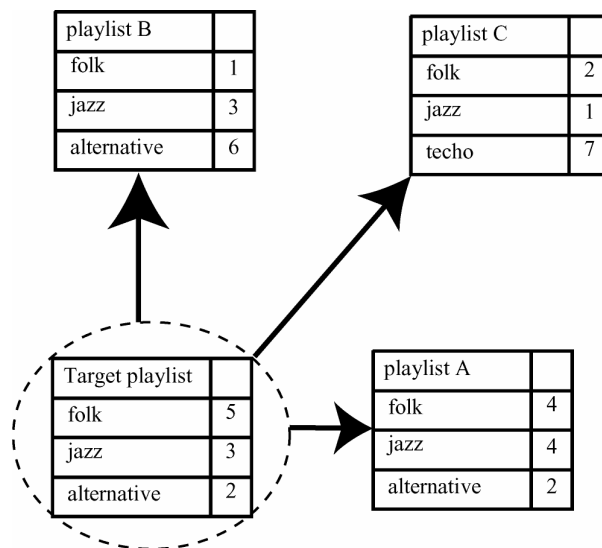


Figure 5.6: A target playlist and three playlists with differing degrees of similarity

5.3.4 Feature Weights

When calculating playlist similarity we apply two weights to these features: a *feature type weight* and a *feature query weight*.

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 result set.

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. This subjective notion of similarity in CBR has been termed *utility* (Bergmann et al. 2001). Stahl (2001) and Branting (2003) have proposed some techniques for determining local utility 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.

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* is given by calculating the proportion of the feature type that is represented by each feature. For instance, if we consider the `genre_` feature type, the *feature query weight* of the `genre_17` feature is 3/10, where the denominator is the total value for the `genre_` feature types in the target case. Thus, the *feature query weight* for feature i of type t can be given as

Equation 5.3

$$wf_{t,i} = \frac{|f_{t,i}|}{\sum_{j \in t} f_j}$$

where $|f_{t,i}|$ 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*.

Equation 5.4

$$ow = \text{feature type weight} * \text{feature query weight}$$

Accordingly, for the playlist case in Table 5.3 the similarity weights for each feature are shown in Table 5.4. In this example not all the `artist_` features are shown.

Table 5.4: The table illustrates how weights are calculated on a per query basis

feature	feature type weight * feature query weight	Overall Weight
genre_1	$0.7 \times (1/10)$	0.07
genre_11	$0.7 \times (2/10)$	0.14
genre_17	$0.7 \times (3/10)$	0.21
genre_7	$0.7 \times (4/10)$	0.28
artist_1201	$0.3 \times (1/10)$	0.03
artist_1118	$0.3 \times (2/10)$	0.06

Similarity metric

The similarity measure we use is given by Equation 5.5. This measure, which is a modified form of a similarity measure known as the *weighted city block* measure (Equation 5.6), was chosen because it works well in matching cases where missing values occur. The commonly used Euclidean based measure shown in Equation 5.7 was not used because it tends to privilege cases with missing values above cases with full case descriptions. In the Smart Radio system the target case defines the features required in each query. Thus the features given by the case in Table 5.3 are those that will only be considered during retrieval. 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 is based on the proportion of the playlist which contains the genre features specified by the target (see Equation 5.8). 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. An example of the similarity calculation is given in the next section.

Equation 5.5

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

Equation 5.6

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

Equation 5.7

$$\text{sim}(A, B) = 1 - \sqrt{\sum_{i=1}^p (a_i - b_i)^2}$$

Equation 5.8

$$c = \frac{|\text{number_tracks_with_target_genres}|}{|\text{total_number_of_tracks}|}$$

Example similarity calculation

We use the playlist illustrated in Figure 5.5 entitled *Hazy Jane Tunes* (and reproduced again in the top left of Figure 5.8) as our target case. We carry out a similarity-based ranking of the three playlists shown in Figure 5.8. The key features for retrieval are the `genre_` features. In this retrieval scenario playlists that match the *folk*, *r'n'b/soul*, *jazz* and *blues* composition of this playlist will score highly. Each playlist is converted to a case representation illustrated by the CBML excerpts in Table 5.6. In order to make our example as clear as possible, the `artist_` features are not represented in this figure. Using the weights calculated in Table 5.4, the similarity scores for each playlist are listed in Table 5.5.

Table 5.5: The ranking of the three playlists according to similarity to the target playlist, playlist 930

Rank	Playlist	genre_ features matched	Percentage of playlist composed of target genre_ features	Score adjustment (c)	Similarity Score	Adjusted Score
1	949	4/4	100	1	0.74	0.74
2	962	3/4	100	1	0.5	0.5
3	964	3/4	80	0.8	0.6	0.48

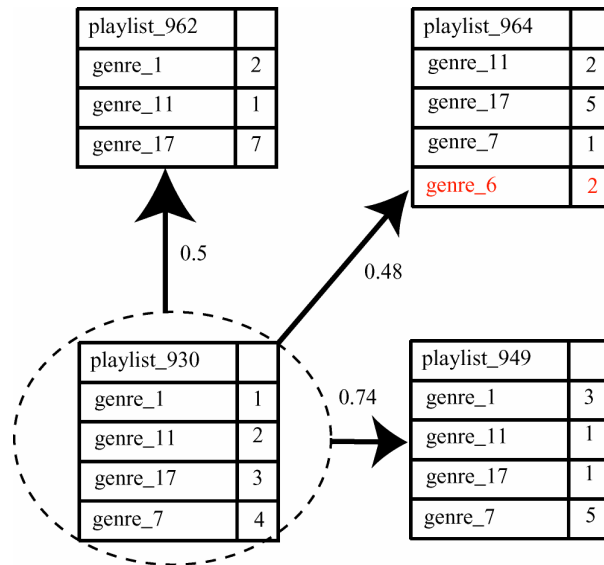


Figure 5.7: The figure illustrates the similarity between the target playlist and three other playlists

Figure 5.7 illustrates the need for a similarity adjustment where a similarity score between cases is based on a partial match. While playlist 962 is missing the target feature `genre_17`, the case is composed solely of features given by the target case. Playlist 964, however, while missing a target feature, `genre_1`, also contains an extra feature not specified by the target, `genre_6`. Given that playlist 964 contains 8/10 tracks described by the target `genre_` features, we apply a similarity score adjustment of 0.8. Without this adjustment, playlist 964 would outrank playlist 962, which in terms of overall playlist similarity, is a better match for the target.



Figure 5.8: The target playlist and three playlists represented in terms of their `genre_` compositions

Table 5.6: The four playlists represented as CBML cases (the `artist_` features are not shown)

<pre> <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> </attributes> </casedef> </case> </pre>	<pre> <case> <casedef casename="playlist_949"> <attributes> <attribute name="genre_1">3</attribute> <attribute name="genre_11">1</attribute> <attribute name="genre_17">1</attribute> <attribute name="genre_7">5</attribute> </attributes> </casedef> </case> </pre>
<pre> <case> <casedef casename="playlist_962"> <attributes> <attribute name="genre_1">2</attribute> <attribute name="genre_11">1</attribute> <attribute name="genre_7">7</attribute> </attributes> </casedef> </case> </pre>	<pre> <case> <casedef casename="playlist_964"> <attributes> <attribute name="genre_11">2</attribute> <attribute name="genre_17">5</attribute> <attribute name="genre_7">1</attribute> <attribute name="genre_6">2</attribute> </attributes> </casedef> </case> </pre>

5.4 Integrating Context Ranking and ACF

Integrating the ACF procedure and similarity-based context ranking requires weighing up a number of factors. Burke (2002) 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. However, the information it yielded was not particularly rich, as obtaining content description for music is particularly difficult, as we discussed in Chapter 2.

5.4.1 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 weaknesses in the ACF strategy used in version one of the system. This primarily involved the insensitivity to user-context which we described in Section 5.3. For this reason, the content-based strategy was always designed as an augmentation to 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 (see Section 4.10.1). 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. Like Smart Radio, *EntréeC* used a single recommendation strategy (in this case content-based) for a considerable time until it was augmented quite recently by the secondary ACF technique. The Smart Radio system, on the other hand, uses ACF as its primary recommendation strategy, which is then refined by a content-based process. As a result of this, Smart Radio is an *automated*

recommendation system whereas EntréeC requires the user to explicitly enter the terms of their restaurant requirements.

The Smart Radio 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.

5.4.2 MAC/FAC

Gentner and Forbus’ 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 CBR 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 an initial set of results. The second stage (FAC) involves similarity-based ranking of this result set according to the user-context.

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 5.9, 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).

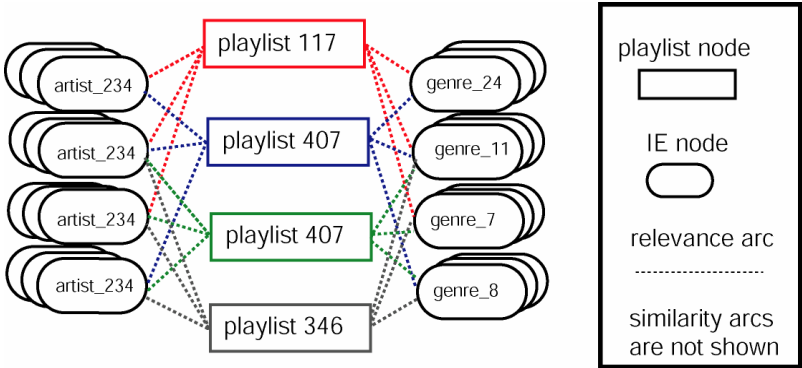


Figure 5.9: A schema of the playlists indexed using a case retrieval net

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. Retrieval involves a spreading activation process through the primed subset of the case retrieval net. The playlists with the 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.

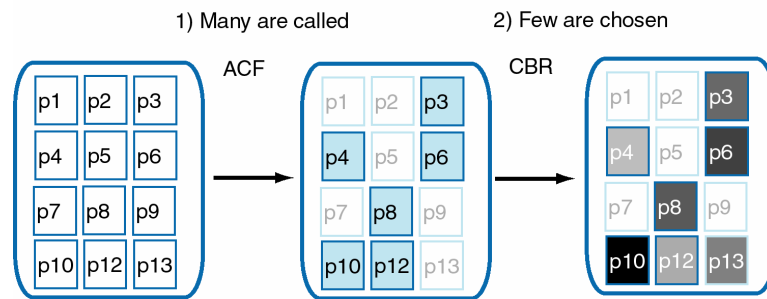


Figure 5.10: The darker shaded cases in the third stage indicate cases which best match the listener’s current listening context

5.4.3 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 from 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. In Chapter 7 we describe how we amend this presentation strategy when we are evaluating the efficacy of our context-based recommendations.



Figure 5.11: The top five playlists at any time are context-sensitive playlists

5.4.4 A Sliding Window to Cater for Negatively Rated Playlists

Up to this point we have suggested that the context-based ranking depends upon the contents of the most recently played playlist. However, in order to cater for negatively rated items in a playlist, the context analyser module uses the previous 10 non-negatively rated items the user has played. This is to avoid the situation whereby the context ranking process presents several more playlists similar to a playlist that the user did not particularly enjoy. In the case where a playlist contains negative ratings, the context analyser fills their place with positively rated tracks taken from the previous playlist. In effect this is a sliding window of size 10 on the previous non-negatively rated tracks. We discuss this issue again in the next chapter.

5.4.5 Recommendation Timing Issues

When the user logs into Smart Radio his/her recommendations are recalculated. These recommendations are then stored in the Smart Radio database. New recommendations are recalculated every 20 minutes for the online user, if he/she has issued feedback to the system. This feedback may be explicit rating of artists/items, playing a playlist or changing a user preference setting. In the intervening period, recommendations are loaded from the database. When a user plays a playlist the recommendation context-sensitive ranking process is triggered and the playlist CRN is primed with case indexes extracted from the recommendation database. As described before, retrieval takes place only over the primed cases in the case memory. This process is relatively inexpensive and can be performed synchronously. Thus the re-ranking of recommendations prompted by the playing of a playlist is immediately available to the user. The next chapter will deal with recommendation timing issues in greater detail.

5.4.6 Refractory Period

Earlier in this chapter we described a user preference, *novelty*, that directly affected the amount of new music the user would find in his/her playlist recommendations. However, even if a user specifies a low novelty setting, we make the assumption he/she does not want to receive playlists with the *same* tracks within the current time frame. There is a danger that this could occur with the context ranking process whereby the most similar playlists to the currently playlist may contain many of the same tracks. We employ a refraction technique whereby recently played tracks are flagged and playlists containing four or more are removed from the recommendation set after the ACF process. The refraction period used in Smart Radio is 3 hours. The effect is similar to the artificial refractory period used by some Neural Networks where the same neurons are inhibited from repeatedly firing within the same time period (McCulloch & Pitts 1943). Once the refractory period has passed, these neurons (and the flagged tracks in Smart Radio) are free to be used again.

5.5 Dealing with ACF Latency

In Section 4.9 we discuss the recommendation latency problem inherent in ACF recommendation systems. In order to stop the filter ‘running dry’, users must have another means of ‘discovering’ items of interest, other than through the ACF filter. An example of this problem in Smart Radio is that the ACF recommender will not recommend new playlists compiled with tracks that have not yet been rated by any other listener. The next three sub-sections describe techniques used in Smart Radio to counter this problem.


5.5.1 Selecting Trusted Neighbours

Prior to the development of ACF systems, social filtering techniques, such as those briefly described in Section 4.1.1, relied upon users picking trusted sources of information. In the *Lens* system (Malone et al. 1987) and the Tapestry mail filtering system (Goldberg et al. 1992) described, users could choose to receive content from nominated users. The Tapestry system assumed that users worked in a small environment and thus could identify other users from which to receive relevant content. In Smart Radio, we provide a means of automatically identifying other users with a similar taste in music. Users can then select the neighbours from whom the system should automatically display newly compiled playlists. Figure 5.12 shows the *New Playlists* panel in Smart Radio in which users can view new playlists compiled by their selected neighbours. Thus playlists composed of previously unrated tracks can be ‘bootstrapped’ into the system.



Figure 5.12: The new playlists panel in Smart Radio

5.5.2 Querying by Playlist

Each playlist panel contains a *query by playlist* icon . By clicking on this icon Smart Radio users send a query to the recommendation server to return a set of playlists similar to the one they are currently viewing (see Figure 5.13). The same similarity metrics, weights, case representation and case memory described in Section 5.3.3 are used for these queries. The *query by playlist* task is not subject to the same constraints as the context task, and will return all playlists similar to the target chosen by the user, irrespective of whether portions of these playlists have been played recently. This facility allows listeners to zoom in on portions of the playlist case base of particular interest to them. Moreover, this facility is independent of the ACF process and can return unrated playlists that would otherwise be passed over by the ACF module.

playlists similar to A Sound Man ...

73 playlists returned: 1 - 10

Playlist	Creator	matches
1. ribsmac's list 1..	ribsmac	0.94
2. rrock	coylel	0.756
3. ribsmac's list 2..	ribsmac	0.756
4. ribsmac's list 2..	ribsmac	0.748
5. poprock1	coylel	0.736
6. ribsmac's list 0..	ribsmac	0.73
7. doyledp's list 2..	doyledp	0.716
8. Conchuir's list ..	Conchuir	0.716
9. Conchuir's test1	Conchuir	0.712
10. fionae 16 March ..	fionae	0.68

[next 10](#)

Title: ribsmac's list 13-May-03 16:58
Creator: ribsmac

Track	Artist	Rate these!
Airbag	Radiohead	☹ ☹ ☹ ☹ ☹
Justice Aversion	Smog	☹ ☹ ☹ ☹ ☹
Ironic	Alanis Morissette	☹ ☹ ☹ ☹ ☹
Virtual Insanitiy	Jamiroquai	☹ ☹ ☹ ☹ ☹
Lovefool	The Cardigans	☹ ☹ ☹ ☹ ☹
Grace	Jeff Buckley	☹ ☹ ☹ ☹ ☹
Torn	Natalie Imbruglia	☹ ☹ ☹ ☹ ☹
A Marriage Made In Heaven	Tindersticks	☹ ☹ ☹ ☹ ☹
With Or Without You	U2	☹ ☹ ☹ ☹ ☹
Break On Through (To The O..	The Doors	☹ ☹ ☹ ☹ ☹

Figure 5.13: The result set from a 'query by playlist'. The top playlist is displayed by default in the right panel.

5.5.3 Explicit Search

Finally, listeners can explicitly search for playlists by stating three of the artists, genres or track names they would like to have included in each playlist in the result set (see Figure 5.14). The system will return a ranked list of playlists with the top list containing the playlist with the maximum number of tracks satisfying the query terms (see Figure 5.15). As with the *query by playlist* request, the ACF engine is bypassed and all playlists in the system that satisfy the query terms are returned.

Search for PlayLists

containing tracks:

By:

Selection 1: i.e. bowie

Selection 2:

Selection 3:

Figure 5.14: The playlist query panel in Smart Radio

Artists: dylan, springsteen, u2

188 playlists returned: 1 - 10

Playlist	Creator	matches
1. monobono's list ..	monobono	3
2. petrolhead's lis..	petrolhead	3
3. petrolhead's lis..	petrolhead	3
4. bee music	BeesKnees	2
5. aidofitz 22 Octo..	aidofitz	2
6. The New Ibiza Mix	jonathan	2
7. rabauke 07 July ..	rabauke	2
8. pippobaudo 19 Ap..	pippobaudo	2
9. test 01 February..	Gregory	2
10. jonathan 14 Dece..	jonathan	2

next 10

Title: monobono's list 14-Feb-03 13:55
Creator: monobono

▶
Ⓜ
⌂

Track	Artist	Rate these!
Ⓜ I Still Haven't Found What..	U2	☹️ ☹️ ☹️ ☹️ 😊
Ⓜ With Or Without You	U2	☹️ ☹️ ☹️ ☹️ 😊
Ⓜ Streets Of Philadelphia	Bruce Springsteen	☹️ ☹️ ☹️ ☹️ 😊
Ⓜ Born To Run	Bruce Springsteen	☹️ ☹️ ☹️ ☹️ 😊
Ⓜ Born In The U.S.A.	Bruce Springsteen	☹️ ☹️ 😊 ☹️ 😊
Ⓜ All I Need	Air	☹️ ☹️ ☹️ ☹️ 😊
Ⓜ Lovefool	The Cardigans	☹️ ☹️ ☹️ ☹️ 😊
Ⓜ Blowin In The Wind	Bob Dylan	☹️ ☹️ ☹️ ☹️ 😊
Ⓜ Lay Lady Lay	Bob Dylan	☹️ ☹️ ☹️ 😊 ☹️
Ⓜ Like A Rolling Stone	Bob Dylan	☹️ ☹️ ☹️ 😊 ☹️

Figure 5.15: A result set for the playlist query: 'artist: dylan, springsteen, u2'

Adaptation

In Chapter 3 we reviewed the CBR cycle. A key step in this cycle is the idea of case adaptation, where a retrieved case, or a combination of retrieved cases is modified to fit the current problem scenario. Adaptation may be automated, in which case domain knowledge is required to produce accurate adaptation rules, or it may be performed manually by the user. In either situation the main idea is the recognition that retrieved cases will not always exactly fit the problem situation.

As we discussed in Chapter 3, adaptation is a problematic area for CBR, as it generally requires some first principles reasoning. Accordingly, many applied CBR systems allow the user to perform any required solution changes. In the Smart Radio system we choose to allow users to manually adapt recommended playlists for a number of reasons. Firstly, an automatically adapted playlist could no longer be represented as a selection of music compiled by a Smart Radio user. If the adaptation rules are too strong, they may completely over-ride the creative input of the original compiler, becoming rudimentary rules for assembling playlists without human help. Furthermore, it is clear that adaptation rules would have to be derived for each user. These could be specified explicitly such as 'swap *jazz* tracks for *electronic* tracks', or 'always remove tracks by *Michael Jackson*'. However, whilst the identification of unsuitable material may seem relatively easy, the criterion for replacing these tracks is less clear. In the case of the first example, a second rule would have to be specified to choose a suitable replacement from the several hundred electronic tracks in the system. In terms of automated adaptation a strategy that might be adopted would be the use of association rules to suggest replacement tracks based on the tracks remaining in the playlist (Agrawal et al. 1996). However, we currently view the automated adaptation phase as a future development for the Smart Radio project.

Manual adaptation

As an aide to manual adaptation we flag tracks that the user may wish to remove before playing a playlist. For instance, even though we use a refractory technique to remove playlists that contain tracks that have been recently played by the listener, playlists containing a few recently played tracks can still make it into the set of recommendations presented to the user. We mark these as shown in Figure 5.16. Using the *edit* function the listener can quickly replace flagged tracks with another by the same artist, or another within the same genre (Figure 5.17).

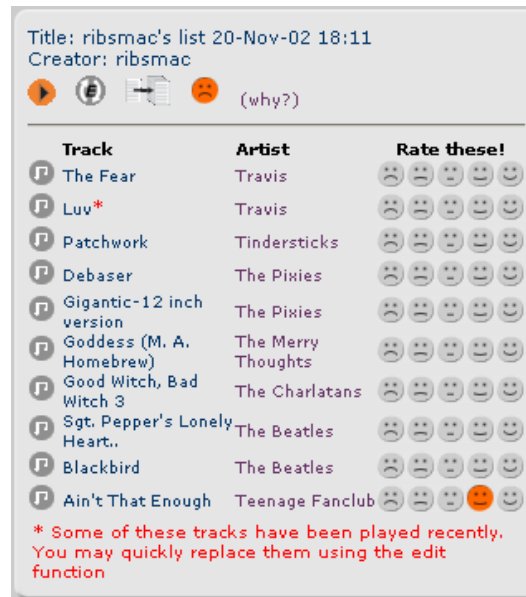


Figure 5.16: Playlist marked for manual adaptation

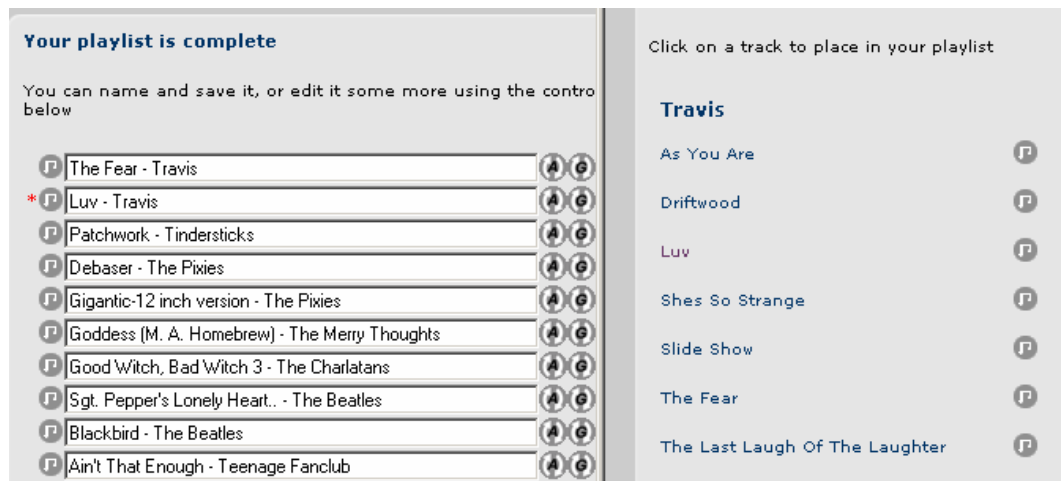


Figure 5.17: Using the edit function the listener can replace flagged tracks with another by the same artist or within the same genre

5.5.4 Fall Back Strategies

In the event of failure of the recommendation strategies outlined above, Smart Radio has three content-based fall back strategies listed below. These strategies are activated in the order given below. Once a set of playlists has been obtained the fall back strategy programme is exited.

- 1) If the user has played a playlist recently, then he/she is offered a set of playlists similar to the target playlist.
- 2) If the user has positively rated some artists, then he/she is offered a set of playlists containing their favourite artists.
- 3) If the user has positively rated some tracks then he/she is offered a set of playlists containing their favourite tracks.

Finally, if no playlists have been found, the user is offered the favourite playlists currently being listened to in the system.

5.6 Conclusion

In this chapter we have described the key techniques applied to the recommendation process in Smart Radio. One of our primary concerns was to leverage the light content description available to us to improve the ACF recommendation process. In doing so, we have implemented a novel hybrid recommendation system in which lightweight case-based strategy provides context-sensitive ordering to the recommendations provided by the ACF module. One of the difficulties in this domain is acquiring content description to mark up cases. Rather than describing individual songs in terms of their most salient features, our case representation describes the *composition* of a playlist in terms of the basic meta-tags we have been able to extract from the mp3 files. The semantic captured by the playlist representation is the mixture of types of music in the playlist. Although features such as *genre* are not particularly discriminating, our view is that these feature mixtures may capture at the *surface* level the intention of the playlist compiler. Thus, *similarity*, in this scenario, refers to how alike playlists are in terms of their composition.

Rather than generalizing a short-term user profile, we use the most recent played playlist as the basis to rank the recommendations produced by the ACF module. In this way we do not have to develop a separate user-profile representation that is compatible with the playlist case representation. The context ranking procedure is triggered when the user chooses a playlist to play. The effect is to issue a “*more like this*” request to the recommendation server. ACF recommendations are thus ranked according to the similarity to the most recently played playlist and the top five displayed to the user. This technique is implemented using a novel MAC/FAC strategy in which the ACF module primes the relevant nodes of a Case Retrieval Net, and activation based on the target playlist spreads only to those cases that have been primed. Although our CBR strategy is of the weak variety, as described in Chapter 3, we feel this is completely appropriate for an extension to ACF which is often used in domains where content description is scarce.

We have also explained three other techniques for working with music content. These techniques are influenced by our view that a recommendation system must cater for how people use the assets being recommended. The *novelty* weight allows users to bias the recommendation engine towards their preference for new music. The playlist re-use window allows previously heard playlists to be played again, if the system finds that the predicted score of the top rated playlist is too low. A *refractory* period prevents playlists containing recently played tracks from being offered to the user. ACF latency is addressed using three techniques: *trusted neighbour filtering*, *querying by instance* and *query-based search*. We show how Smart Radio users can manually adapt playlists using a simple user interface.

In the next chapter we describe the system and data architecture used in Smart Radio. In particular we will describe how data is regularly compiled to update the recommendation modules, and our novel implementation of the ACF module as a Case Retrieval Net.

Chapter 6: System and Data Architecture

In this chapter we describe the overall architecture of the Smart Radio system. We will be concerned throughout with the processes of collecting and refining data in order to update the models that drive the recommendation process. We present our Data Manager module and explain our algorithm for deriving implicit track and artist scores. We discuss our technique for bootstrapping new users into Smart Radio using artist scores rather than track scores. We demonstrate that Smart Radio is reliant upon the input of a few prolific playlist compilers, and we suggest that the role of *community leader* is the primary incentive for such users.

We describe the operation of our recommendation server, and explain our novel implementation of a Case Retrieval Net (CRN) for ACF. We illustrate how computation is distributed through the nodes of a CRN during query time. We conduct an experiment that compares CRN neighbour retrieval with a flat neighbour search and find that the CRN retrieval is considerably faster.

6.1 System 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 users' current interests (see Figure 6.1).

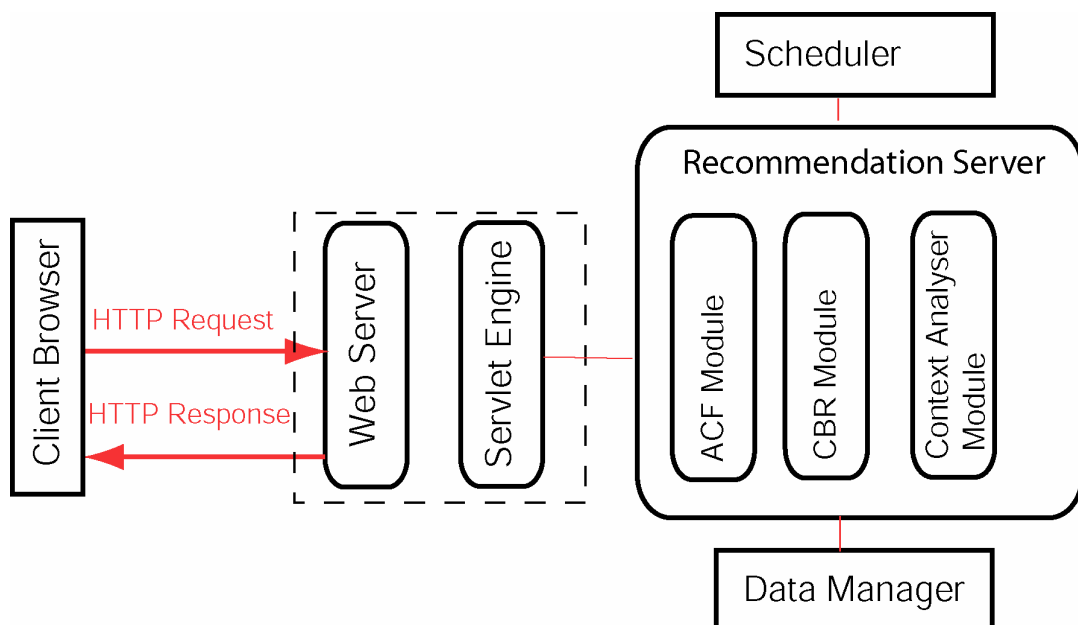


Figure 6.1: The primary components in the Smart Radio system

The server side components were written completely in the Java™ language. We choose Java because of its strong support for distributed server side programming³⁷ and the ease with which it can be deployed on different platforms. In the following section we will discuss in detail the interaction of these components.

6.2 Web Service

The communication between the client and the server is carried out using HTTP, the standard web protocol. Smart Radio uses an Apache³⁸ web server as the system *front door* through which client requests are received and responses despatched. The web server also delivers static material such as images, Javascript pages and style sheets.

The web server works in conjunction with a servlet engine which handles the processing logic required by requests that require dynamically generated content such as a web page displaying the user's most recent recommendation set. The servlet engine is a multithreaded container designed to provide a processing facility in the Java language³⁹ for request/response protocols such as HTTP⁴⁰. Each servlet is a Java programme written to deal with the logic processing required by a specific request. For instance, in Smart Radio the *recommendation* servlet will handle a client request for a web page displaying his/her recommendations. The web server recognises that requests are to be passed to the servlet Engine for processing by identifying the request URL as a request for dynamic rather than static content. This is done by reserving a folder name as an identification for servlet processing. In Smart Radio the reserved folder is */smartradio/*. Servlets act as *controllers* selecting which recommendation/database services to call, and choosing which presentation layer (View) component to render.

Figure 6.2 illustrates the system architecture from the perspective of the web service. The red arrows point out the logical path followed by a request from the client browser until a response is received from the web server.

³⁷ <http://java.sun.com/j2ee/faq.html>

³⁸ <http://www.apache.org>

³⁹ <http://java.sun.com>

⁴⁰ <http://java.sun.com/products/servlet/>

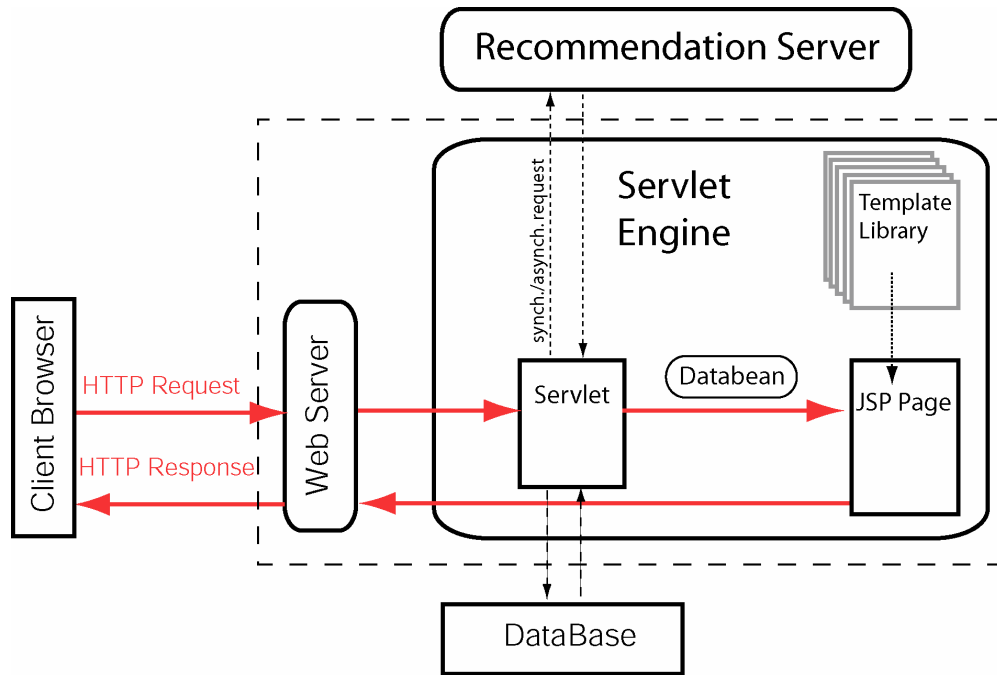


Figure 6.2: The system architecture from the perspective of the web service

Model-View-Controller (MVC)

The client server architecture in Smart Radio follows the Model-View-Controller (MVC) design pattern. The MVC paradigm is a way of breaking an application interface into three parts: *the model*, *the view* and *the controller*. MVC was originally developed to map the input, processing and output roles in GUI design for Smalltalk-80 applications (Goldberg & Robson 1983). More recently it has become an increasingly popular design pattern for client-server systems on the web (Alur et al. 2001). The key to the MVC design pattern is the decoupling of data access, application logic, data presentation and user interaction. In this context the functionality of the MVC is summarised in Table 6.1.

Table 6.1: A summary of the functions of the Model-View-Controller design pattern

	Description	Smart Radio Components
Model	The Model encapsulates the business logic of the application. It can respond to requests and notify the View component of changes, and updates.	The Recommendation Server, Database Access component
View	The View renders the contents of a model. It is the view's responsibility to maintain consistency in its presentation when the model changes.	JSP pages and Data Beans
Controller	A Controller is the means by which the user interacts with the application. A controller accepts input from the user and instructs the model and view to perform actions based on that input. In effect, the controller is responsible for mapping end-user action to application response.	Servlets

The *Controller* logic in Smart Radio is carried out by servlets. This involves checking for new recommendations, creating data containers (data beans), and requesting updates to the recommendation server either asynchronously or synchronously, which we discuss later in the chapter.

Data Beans are data structures created by the interaction of the Servlet with the database or the Recommendation server (the model components). They are created in order to provide content for each Java Server Page (JSP), the HTML rendering component of the servlet engine. In this way they provide an interface between the Model and the View components. A data bean is created for each client request for specific information such as a list of a user's recommendations or the top ten lists in Smart Radio. In order to reduce the number of requests being made to the database, which can be a bottleneck, data beans are cached (by the controller). Servlets will check the cache before requesting new information from the database. When the control logic required for the request has been carried out, the servlet engine forwards the request and a data container, the data bean, to the JSP rendering engine.

JSP pages, which act as the *View* component of the system, are servlet extensions which are designed for rendering the presentation layer in an application. Each page contains HTML in which some Java code is embedded, mainly for providing data access. In the case of Smart Radio, data access is performed by the servlet component and the contents rendered into a data bean. JSP pages format the data in the data bean into HTML pages. We use JSP pages in conjunction with a JSP template library which allows us to standardise and modularise our presentation design.

The *Model* functionality is carried out by the Recommendation server, and the database connectivity module. The Recommendation server can be queried synchronously or asynchronously. We will describe its role in the overall system in greater detail in Section 6.6.

6.3 Scheduling Component

The scheduling component in Smart Radio ensures that the most up-to-date data is in use by the system. It controls the following functions:

6.3.1 Top Playlists Update

The top playlist facility operates using a sliding window of 7 days. Each playlist in this section is ranked in decreasing order of popularity over the previous 7 days with users other than the list creator himself. We decided not to count the playlist creator's playlist usage after we noticed that some users were playing their own playlists several times in a row, apparently in an attempt to gain the number one slot.

6.3.2 Data Recompile

The scheduling component requests the Data Manager to check whether new data has been detected in the database since the last data update. If new data is detected the data manager updates

the ACF data model for those users who have submitted data during the interval. We discuss the Data Manager in more detail in Section 6.4.

6.3.3 Recommendation Server Update

When we have updated the ACF database model we then update the internal data model for the Recommendation Server. This is not a full update; the data model is updated from the ACF database for users who have recently submitted feedback. We describe the internal data model of the server (a case retrieval net) in Section 6.6.2.

While the update is being carried out the server is briefly not available for synchronous requests, in which case these requests may time-out. Asynchronous requests are queued and can be executed when the update is completed.

6.3.4 Recommendation Update

The recommendation service in Smart Radio is designed to handle synchronous and asynchronous update requests.

Synchronous requests are those requests where the response from the recommendation server must be co-ordinated with the response by the web server. These are required for on-demand recommendations. We use this service when we bootstrap new users, and when we carry out context-based ranking in response to a playlist event.

Asynchronous requests are requests where a timely response is not required. We use this service for our periodic updates of the recommendation database.

6.4 Data Collection and the Data Manager

We rely upon the data we collect from our listeners to make recommendations in Smart Radio. Listeners can provide explicit ratings on individual music tracks or individual artists on a scale of 1–5, where 5 is the top score. We also log implicit feedback, where the user has listened to music items but has not explicitly rated these items. In this case we allocate positive ratings to frequently played items, and negative ratings to less frequently played items. We discuss the algorithm for deriving implicit data later in this section.

The data model of the ACF recommendation component is regularly updated so that recommendations can be made with the most up to date data available. Since feedback may be posted by several users concurrently, we use a multi-threaded data logger, similar to that used to log requests in a web server. The Data Manager is the component that periodically converts raw log data collected by the system into data that can be used in the Recommendation engine. The data conversion period can be adjusted. Currently the database is checked for new data every 2 minutes, if users are on-line. The Data Manager does this by checking a database table, `new_data_flag`, which contains records indicating which users have posted new data since the last data conversion. This table records the time and the *type* of data that has been posted by the user. The Data Manager

can then make an incremental update to the ACF database for each user profile listed in the `new_data_flag` table.

As we saw in Chapter 4, the data used in ACF systems usually consists of three columns – `userId`, `itemId` and `score`. Smart Radio has two such databases, `recdata_item` and `recdata_artist`, which store the data driving the ACF algorithms. Rather than allowing multiple user threads to access these key ACF databases, the data manager periodically compiles new log data into the ACF format and updates the central ACF databases. The Data Manager is also responsible for deriving the implicit scores for item and artist data.

6.4.1 Implicit Track Data

As we discussed in Chapter 4, assigning an implicit value to a user action is likely to be an imprecise affair. Different heuristics have been devised as indicators of user interest such as the time spent reading a piece of text (Morita and Shinoda 1994, Konstan et al. 1997, Rafter & Smyth 2000). In Smart Radio we make use of the fact that very often people will listen to music they like repeatedly, and listen less often to music they dislike. We measured the correlation between the ratings and the number of listens for each user, and found the mean to be 0.236. Although this is a lower correlation than we might have expected it can be explained by noise in the count of the number of listens. Since Smart Radio delivers compilations of music, a compilation containing an item that the user dislikes and has rated negatively may be played frequently because of other more favourable items in the compilation.

We calculate the mean and standard deviation of each user’s rating set and listening count set. Our implicit scoring algorithm allocates positive scores to items the user has listened to above his/her average listening count, and negative scores to items the user has listened to below the average listening count. Each score is allocated around the user’s explicit mean score. A positive vote is calculated as the mean plus a single standard deviation, a negative vote is calculated as the mean minus a single standard deviation.

In certain cases we have no explicit data on which to base implicit track scores. In such situations we allocate the average explicit score in the database for each item, if the count for that item is above the user’s average item count. This allows us to provisionally correlate the user with other users for the purpose of making recommendations. However, we do not use this user’s data for making recommendations for other users. As such, in Chapter 7 the implicit track data we evaluate is based on user data where users have also submitted explicit scores.

6.4.2 Implicit Artist Data – Dimensionality Reduction

In Chapter 4 we discussed dimensionality reduction, a technique for reducing dataset sparsity where the aim is to reduce the *horizontal* dimensions of the user–item matrix. Singular Value Decomposition (SVD), a matrix factorisation technique, has been used for this purpose (Sarwar et al. 2000a, Billsus & Pazzani 1999). SVD is the basis of latent semantic indexing (LSI), a technique

used in Information Retrieval to solve the problems of *synonymy* and *polysemy* in a corpus of documents (Deerwester et al. 1990, Berry et al. 1995). When SVD is used in an information-rich environment like information retrieval, it is capable of producing *reduced* features that still have human level semantics attached to them. With data that has very little semantic information attached to it such as in an ACF environment, SVD is used simply as a matrix factorisation technique – in which case the reduced feature set has no human level semantics associated with it.

While reducing the dimensionality of the user–item matrix in order to improve ACF performance is desirable, so too is the goal of producing an interpretable feature set, particularly for the purpose of *bootstrapping* a new user into the system. Since the Smart Radio dataset has very little content attached to it, we did not consider SVD to be a solution that could meet both objectives. However, on closer inspection we realised that the content data available to us did provide us with the type of categorical information we would be looking for from a LSI perspective. The content ‘features’ associated with our music data can be used to describe each music track as belonging to an *artist* and *genre* class. Therefore we decided to reduce the *user–music* item matrix by mapping it to a *user–artist* matrix. To calculate a score for a user–artist we simply average the scores for tracks from that artist that the user has explicitly rated.

Equation 6.1

$$S_{ua} = \text{Average}(t_{ui})$$

$$t_{ui} \in A$$

In Equation 6.1, A represents the set of tracks by artist a . The score allocated to user u for artist a is the mean of the explicit ratings given to tracks, t_{ui} , by artist a by user u . Applying Equation 6.1 to the explicit track data we obtain a transformed dataset with the same number of users but with the item dimensions reduced from 4131, the number of music items, to 333, the number of artists represented in the dataset. This decreases the sparsity of the dataset from 0.9734 (`track_explicit`) to 0.8682 (`artist_implicit`). We perform a detailed analysis of this dataset in Chapter 7. In the next section we will discuss the issue of bootstrapping in ACF which concerns the performance of the algorithm where there is a deficit of appropriate data. As we will demonstrate, the user–artist dataset is useful for bootstrapping new users into the system.

6.5 A Short History of Bootstrapping in Smart Radio

The first Smart Radio system was a barebones affair. Users could assemble playlists by selecting tracks that were indexed by genre. The issue of cold-starting the ACF recommendation engine was addressed initially in a very informal manner, and relied upon the goodwill and patience of colleagues. An email was sent to the staff and postgraduate students in the Computer Science Department, encouraging them to log-on and compile playlists. Many did, although, at this stage, few recommendations were available because of the lack of data in the system. We found that the

‘churn’ rate was very high, with users logging on and leaving after submitting very little rating data. This demonstrates a second barrier to starting ACF systems, which is rarely mentioned in the literature – lack of user incentive. Avery and Zeckhauser (1997) suggest that there is little reason for the user to provide early ratings seeing that this information will not perceptively improve his/her recommendations.

With very little data the early Smart Radio System had difficulty in making personalized predictions. We can see this by simulating *early-stage* ACF on the Smart Radio dataset `track_explicit_data` (which we describe in detail in Chapter 7). In this experiment we use the 71 user instances in the dataset, but withhold 90% of each user’s data. Using the *Leave_One_Out* technique (explained in Chapter 7), we obtain a measure of Mean Absolute Error per user. We increase the amount of data in each user’s profile by 10% and repeat the experiment until we are using 100% of the data available. The experiment simulates the performance of the ACF algorithm as users gradually increase their amount of ratings over time. As we will discuss further in Chapter 7, the problem in early stage ACF is data sparsity, which implies that there is insufficient data on which to calculate correlations with other users. The results of our experiment are presented as a graph in Figure 6.3. We also include the results from a baseline ‘averaging’ technique (explained in Chapter 7). This technique is used as a measure as to whether our ACF strategy can make personalized predictions or not. As we can see from the graph, both techniques suffer due to lack of data. The ACF strategy is still capable of making predictions with less error where there is a low percentage of data available. However, the graph would suggest that the output from the algorithm is unstable where there is little data available, and only begins to stabilise from the 60% level upwards.

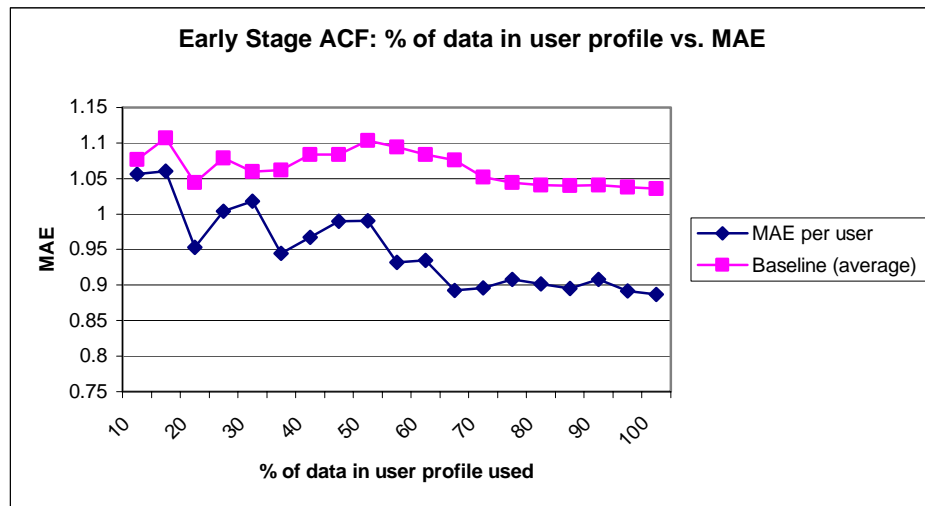


Figure 6.3: A demonstration of the problem of using ACF when users have not yet rated many items

Despite the poor performance of the recommender system in the first few months of its deployment, several users persevered and built up considerable portfolios of playlists. The

persistence of these users, and their reasons for using the system, highlighted a particular aspect of our recommendation scenario which we later enhanced.

Many ACF systems make recommendations without divulging their source. This is difficult if many neighbours have endorsed the recommendation. In Smart Radio, irrespective of the number of neighbours that have contributed to a playlist recommendation, the playlist has one author whose name is publicly available. Thus the Smart Radio listener can identify the author behind each recommended programme. Our aim was to help users to recognise which neighbours they tended to trust when it came to playlist compilation. In this respect, we endorsed the premise of Hill et al. (1995) that a recommender system should promote rather than replace social processes.

It was this facility that attracted a small number of regular, prolific users, who used Smart Radio to swap playlists between each other. With a small number of users and a relatively small number of playlists, a playlist built by one user tended to show up immediately in the recommendation set of the other users. Users could choose to play the list depending upon whether they trusted the user.

After we improved the user interface to Smart Radio, extended the search capabilities and added more content, we acquired a significant amount of new users. The addition of more data improved the performance of the ACF algorithm, with the result that many users continued using the system. However, with the increased number of users and playlists, new playlists by previously trusted sources tended to become lost amongst recommendations provided by newer users. To address this effect we provided a means whereby users could nominate a neighbour as a trusted source, so that new playlists compiled by that neighbour were always available to the user (see Figure 6.4). This is also one means of addressing the latency problem in ACF, whereby new content cannot be recommended until rated by other users.

Your Nearest Neighbours

Using the checkboxes below you can choose the Neighbours whose new playlists you wish to view

Select these Neighbours

	userName	Correlation	Common items	Select
1.	coylel	0.194	124	<input checked="" type="checkbox"/>
2.	ribsmac	0.245	95	<input checked="" type="checkbox"/>
3.	evilator	0.084	95	<input type="checkbox"/>
4.	PadraigC	0.076	88	<input checked="" type="checkbox"/>
5.	ginger	0.166	81	<input checked="" type="checkbox"/>
6.	BeesKnees	0.204	80	<input type="checkbox"/>
7.	oconnonp	0.225	69	<input checked="" type="checkbox"/>
8.	p_clerkin	0.142	64	<input type="checkbox"/>
9.	Happydude	0.228	58	<input checked="" type="checkbox"/>
10.	bradyo	0.231	53	<input checked="" type="checkbox"/>
11.	fionae	0.224	49	<input checked="" type="checkbox"/>
12.	daggerd	0.155	47	<input type="checkbox"/>

Figure 6.4: Users can select those neighbours whose new playlists they wish to view



Figure 6.5: Smart Radio users can view new playlists compiled by selected neighbours

As we discuss in the next section, Smart Radio is still dependent upon the industry of a few prolific users who regularly build new playlists. It is clear that these users view themselves as *community leaders* with respect to the type of music they like.

6.5.1 Community Workers

In Smart Radio, there is an additional *labour* overhead which is not a feature of comparable recommender systems. If Smart Radio only recommended discrete sets of music tracks, the only work we would require from our users would be that they rate (explicitly or implicitly) a portion of them. Given that the playlist is the unit of recommendation in Smart Radio, we require our users to compile new playlists, *as well as* provide us with ratings.

In the Smart Radio system new playlists can be built from scratch or adapted from existing lists. In the following section we make some observations on the problem of relying on users to provide new playlists, without which our recommendation engine will run dry. Our experience in Smart Radio is that many users are passive users of the system relying upon the recommendation engine and search facilities, and, very rarely if ever, compiling new playlists. Other users are prolific, becoming *taste* leaders as such.

6.5.2 New Playlists

Figure 6.6 and Figure 6.7 illustrate data collected during our evaluation period which ran from 08/04/2003 to 17/07/2003. Chapter 7 will provide a much more extensive analysis of this period. The graph in Figure 6.6 shows that the percentage of new playlists played in the system is approximately 20.2% of the 1012 playlists played in total in the system during the evaluation

period. These 170 new lists were built by 29 out of the 58 listeners (50%) during that period, the majority of these being built by a few prolific list makers. We can see this from the graph in Figure 6.7 where we plot the number of users against a series of ranges representing the number of new playlists compiled. 23 users compiled between 1 and 5 playlists while only 6 out of 58 users compiled more than 6 new playlists. It is important to recognise that for a system like Smart Radio to operate, a certain proportion of users must regularly contribute some work so that the community at large can benefit. Given that building a new playlist is not going to improve user recommendations, there is very little incentive to carry out this work. This is similar to the *early-rater* problem discussed earlier. However, since Smart Radio user names are attached to playlists, one incentive that cannot be discounted is simply the prestige that users may feel in becoming arbiters of taste within a community of their peers.

Another perspective on this might be to investigate why some users refuse to contribute to a shared, community-based system. This phenomenon, known as the *free-rider* problem, has been widely studied in the discipline of social science (Sweeney 1973) and more recently in analyses of user behaviour in file-sharing networks (Golle & Leyton-Brown 2001, Adar & Huberman 2000, Saroiu et al. 2001). Certainly similar analyses could be applied to ACF-based systems such as Smart Radio, if only to qualify the parameters affecting the minimum ‘work sink’ required for an ACF system to operate.

Previous research has recognised the key role that a few energetic contributors make to community-based sites (Ferrario & Smyth 2001). In Chapter 8 we will suggest that Smart Radio could easily be extended to include a more formal role in moderating the community for these users.

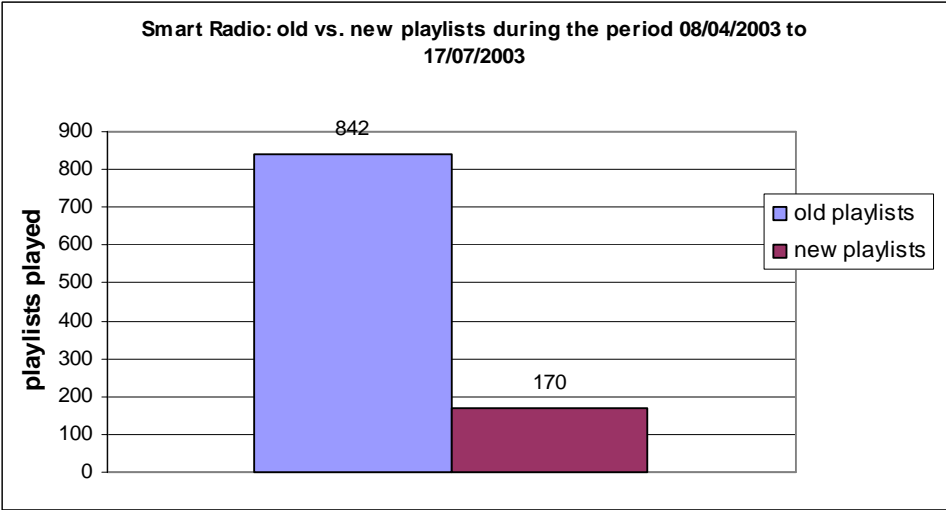


Figure 6.6: An illustration of whether users compiled new playlists or played previously compiled lists

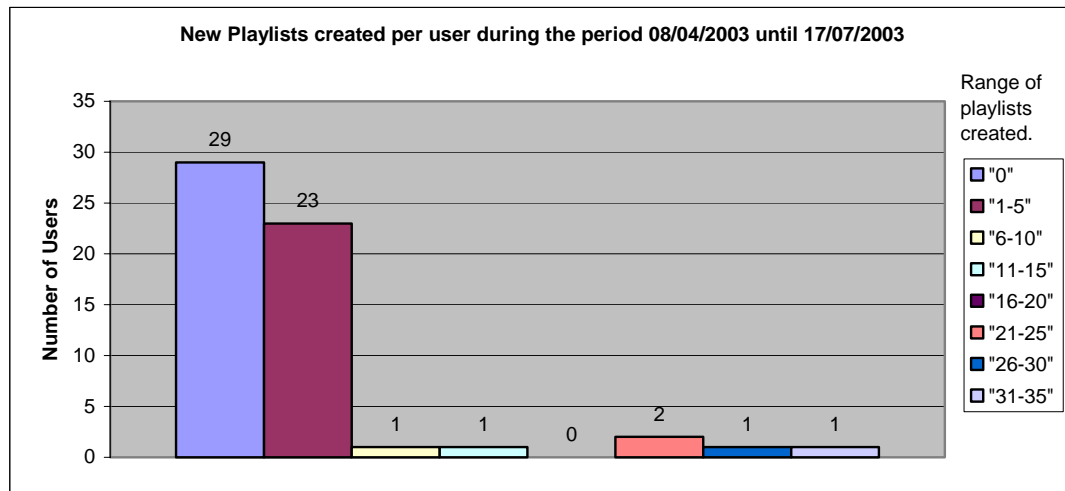


Figure 6.7: Users in Smart Radio rely upon the industry of a few prolific playlist compilers

6.5.3 Bootstrapping New Users

Providing recommendations for new users is a problem because they have not yet built up a rating profile and so cannot be correlated with other users. Since ACF datasets are sparse, users must build up quite a few ratings before they have sufficient data to be correctly correlated with other users. While building up their profile, new users may receive poor recommendations or no recommendations at all. Naturally, many users may not persevere when there is no immediate benefit to them.

In the first version of Smart Radio new users were offered a list of the ten most popular current playlists. Our original idea was that by choosing one of these lists users would be immediately correlated with several other users. However, this facility could not cater towards users with more specialised tastes, which would lie outside playlists within the top ten. During this period Smart Radio suffered from a lot of user ‘churn’ – users logging on and not coming back again. Accordingly we re-implemented our bootstrap strategy. The key objective in bootstrapping a user into the system is to elicit the minimum amount of information necessary so that good correlations can be made with other users. To this end we implemented our dimensionality reduction algorithm which transformed the user–item dataset to a user–artist dataset. We redesigned the user interface of the bootstrap facility so that, on registering with Smart Radio, new users were offered the opportunity to rate a selection of artists, rather than individual music tracks. These artists are indexed by genre, and there is at least one rating in the dataset for each artist. Users may listen to a sample of the artist’s music if they are unsure of what rating to allocate. The key idea is that by asking users to rate n artists, we have a much better chance of correctly correlating them with their neighbours than if we ask them to rate seven music tracks. This is due to the improved sparsity of the user–artist dataset (0.8682). With this we hope to achieve a better result by asking the user to perform the same amount of work.

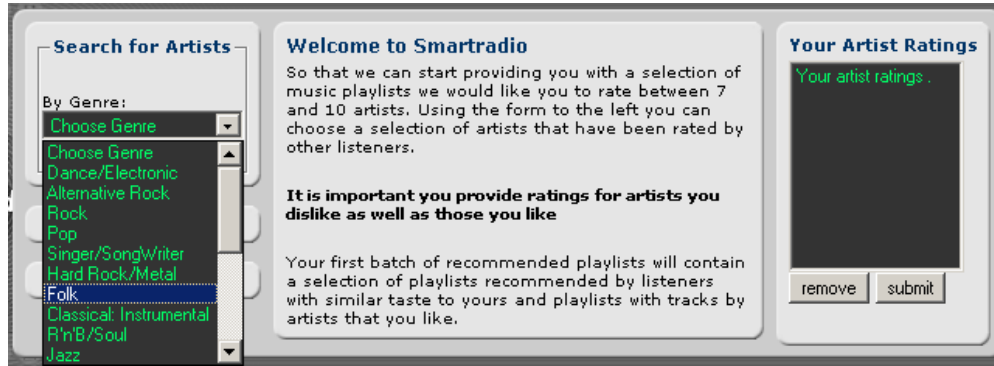


Figure 6.8: On registering, users are encouraged to rate a number of artists so that they can immediately start receiving recommendations

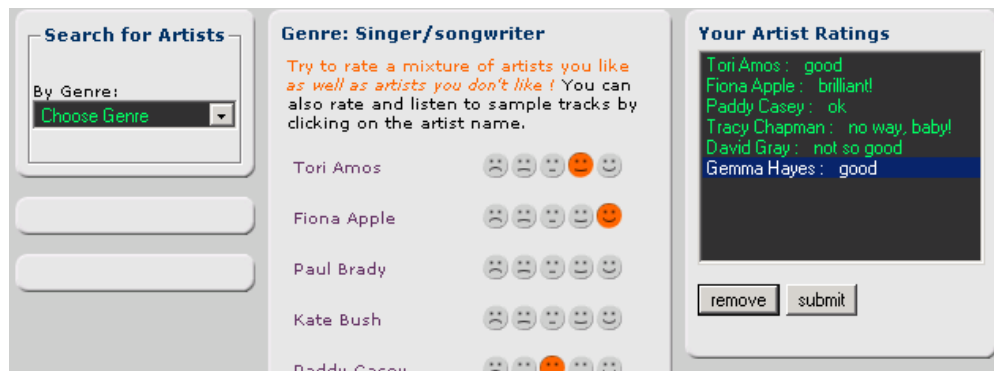


Figure 6.9: During the bootstrap process artists can be chosen from 21 different genres

Unfortunately, we do not have churn data prior to the introduction of the new bootstrap strategy so we cannot say for certain whether we improved the bootstrap experience for the new user. However, we can run a simple off-line experiment that demonstrates the efficacy of the technique.

6.5.4 Bootstrap Experiment: Given- n

In this experiment we explore how n artist ratings during the bootstrap stage may outperform n track ratings. We use a technique called *Given- n* (Breese et al. 1998) which allows us to examine how each dataset performs when there is relatively little rating available for the test user. Applying this technique to the `track_explicit` dataset, we cycle through the dataset, randomly selecting n ratings from each test user as the set of observed ratings. This is the rating set the user will be correlated on. The goal is to predict the remaining ratings in the test user instance. The value for n is initially set at 1 and then incremented until we reach 10. The plot for this dataset is shown by the blue series in Figure 6.10.

Performing this experiment on the `artist_implicit` dataset is a little more complicated. Once again we cycle through the dataset. For each test user instance we select n artist ratings as the set of ratings the test user will be correlated on. Our goal is to make predictions on a subset of the test user track data. Unlike the previous test we have not split the `track` rating data into training and test data. The test data we use, in this case, is the set of track ratings that have not contributed to the ratings for the n artists in the training set. For example, for user U , if $n = 2$, and the artists selected are ‘Bob Dylan’ and ‘Madonna’, then we choose as a test set all tracks rated by user U that are not by Bob Dylan or Madonna. We therefore make predictions on items that have no connection with the training set. However, this means that our test set will be smaller than the test set used for the `track_explicit` dataset. The plot for this dataset is shown by the pink series in Figure 6.10.

On the face of it, the `artist_explicit` dataset appears to have a lower mean absolute error per user when n is low, which is similar to the bootstrap phase for the new user. It would therefore seem to be a better choice of dataset to use when finding neighbours for new users. However, this test is imperfect because the test on the `artist_explicit` dataset uses a smaller test set than the test on the `track_explicit` dataset. We carry out extensive analyses on these two datasets in Chapter 7.

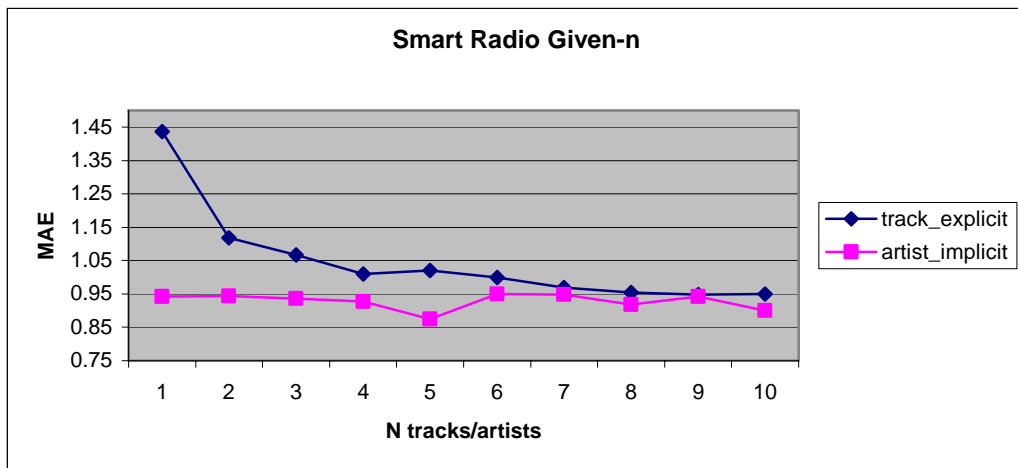


Figure 6.10: The performance of track data vs. artist data during the bootstrap phase using the *Given-n* technique.

6.6 Recommendation Modules

The recommendation engine in Smart Radio is composed of the following components: the ACF Recommendation module, the CBR module and the Context Analysis module. In this section we will concentrate primarily on the data and system architecture of the ACF recommendation module.

6.6.1 A CBR View on ACF

Towards the end of Chapter 4 we observed how the typical ACF approach is similar to *case completion*. As we described in Chapter 3, case completion involves problem solving where the goal is to specify the series of steps in a process that will lead to a solution. Thus the solution part of the case is not differentiated from the problem description part of the case. The solution is derived by gradually elaborating the initially under-specified target case. The case description is usually incrementally built up using a dialog with the user.

At each step of the case completion process, cases are retrieved and case completion information suitable for the target case is extracted and offered to the user. The user selects the pieces of completion information he/she deems suitable for the current process, and the cycle resumes until the case has been fully specified (see Figure 6.11a). In some case completion tasks, the order in which the case is completed is not important. For instance, the *Nodal* system (Cunningham et al. 1998) uses an information theoretic analysis to choose the next feature value to offer the user. However, increasingly dialogue ordering is being recognised as an important part of conversational systems (Aha et al. 1998, Schmitt & Bergman 2001, Bridge 2002). This research recognises that users of dialogue-driven systems expect a sensible ordering to the questions they are required to answer. Also, if the case completion technique describes a process or a plan then there must be constraints to the case completion features available at each step which reflect dependencies between subsections of the task or plan (Bergmann et al. 1998). Therefore, in case completion scenarios we can make a distinction between systems that use surface similarity such as *Nodal*, and systems that involve structural similarity. Surface similarity as defined by Gentner (1983) concerns feature value similarity, while structural similarity is defined using the relations *between* features, such as ordering constraints.

We can usefully compare the ACF process to case completion. The comparison will highlight the similarities ACF has with lazy, similarity techniques such as case completion CBR, but also highlight the deficiencies. Using this analysis we will be able to draw upon techniques used in case completion and apply them to the ACF context.

We can see that the ACF process and the case completion process both involve the incremental elaboration of the target case based on feedback given by the user. An ACF system uses the information it has to hand to retrieve similar user profiles and extract completion information for the case profile which is then offered to the user (see Figure 6.11b).

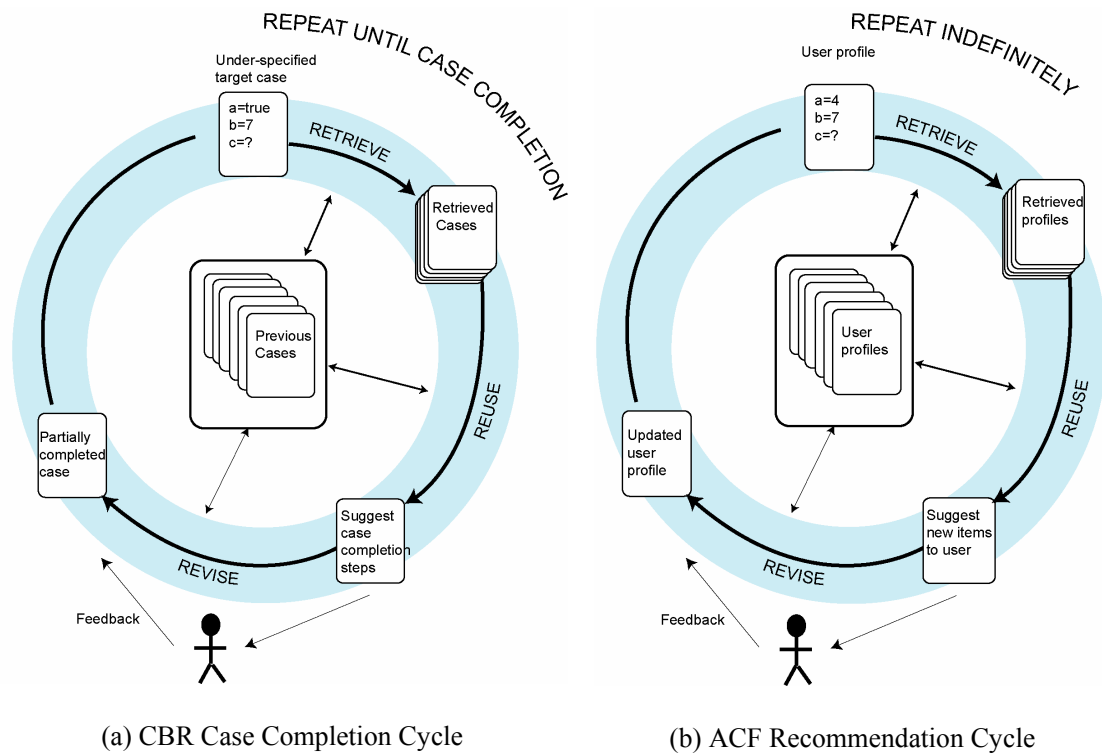


Figure 6.11: A comparison of the CBR case completion process with the ACF recommendation cycle

Negative user feedback may move the user toward a different set of neighbour profiles which is then used to make the next set of recommendations. So both techniques are concerned with the stepwise filling out of the target case.

Many CBR case representations are not concerned with structural similarity, and use surface similarity as their only means of assessing similarity. An ACF profile might be considered such a case representation – it is composed of a number of unordered, feature value pairs each representing an item and the rating assigned by the user. Many case completion processes, however, do have some ordering relations between features since tasks cannot be completed in an ad-hoc way. While the data informing an ACF profile does have an ordering relation (it represents a shallow trace of the user’s ‘consumption’ over time), this relation is not represented in the ACF profile. Since the ACF algorithm uses surface similarity only to assess similarity, any semantics associated with the original ordering are discarded. Thus, unlike case completion entities, ACF case completion entities (ACF recommendations) are presented in an ad-hoc way, without concern for the order of the entities preceding them.

So at a surface level we can recognise the resemblance of an ACF profile to a CBR case, but at a structural level we should note that the ordering information in an ACF profile is absent. This observation leads us to consider how we might make use of ordering information in ACF. CBR systems that have temporal or ordering constraints, such as process planning, require sources of domain knowledge that specify the dependencies between process steps. Bergmann et al. (1998) specify three possible sources – a domain-specific reasoning system, user constraints, and a static

analysis of the domain theory. However, all three techniques presuppose a domain model which is entirely absent from ACF applications. In fact, in most ACF recommenders we have no knowledge of the user, and very little knowledge, if any, of the content being used. It is this shallowness of the knowledge available which makes using ordering relations in ACF profiles unhelpful. In Chapter 6 we used a technique called context-boosted ACF to address this lack of ordering by ranking ACF recommendations according to a simple profile based on the *most recent* ordering relations in the user profile.

6.6.2 Using ACF in a Case Retrieval Net

Our observation that the ACF process is similar to case completion at the surface level leads us to apply a technique used in case completion to ACF. Since the ACF process involves an extended dialogue with many users, the amount of information in the system will increase very quickly. As we discussed in Chapter 4, ACF suffers from the same weakness as all memory-based techniques, namely computational expense at query time and a large memory overhead.

While tree structures have been used to index memory-based instances for efficient retrieval (Bentley 1975, Friedman et al. 1977, Wess et al. 1993), such techniques are unsuited to instances where a large proportion of the feature-values are missing. However, Lenz (1999) and Burkhard (1998) suggest that Case Retrieval Nets (CRN) are suitable for problem situations such as case completion where missing case information is usual and where there is no clear distinction between problem description and solution.

We have already described the structure of a CRN in Chapter 3. Briefly, a CRN is composed of nodes called Information Entities (IE). An IE is a basic knowledge item such as an attribute-value pair. A case is composed of a set of IEs. A CRN consists of a network of unique IE nodes connected to each other via similarity arcs. Each case node is connected to its constituting IE nodes via *relevance arcs*. IE nodes are not duplicated in the Network so several cases may share the same IE node that represents a particular attribute-value pair. Different degrees of similarity and relevance may be expressed by varying arc weights.

A CRN is a memory model that allows for the efficient retrieval of a relatively small number of relevant cases from a huge case base (Lenz et al. 1998). The central idea is to apply a spreading activation process to a net-like case memory in order to retrieve cases similar to a posed query case. Thus, the CRN structure has some similarity to the structures underlying artificial neural nets and associative memory techniques. In contrast to neural networks, however, all the nodes and arcs in that net have precise meaning. Whereas indexing models for instance-based reasoning systems are typically tree structures that allow for a *top-down* search, a CRN enables a *bottom-up* process that attempts the *re-construction* of similar cases (Lenz et al. 1998).

We have implemented our ACF memory structure using a CRN in which the case node represents a user identifier and item-value pairs are represented by information entities. This structure is illustrated in Figure 6.12.

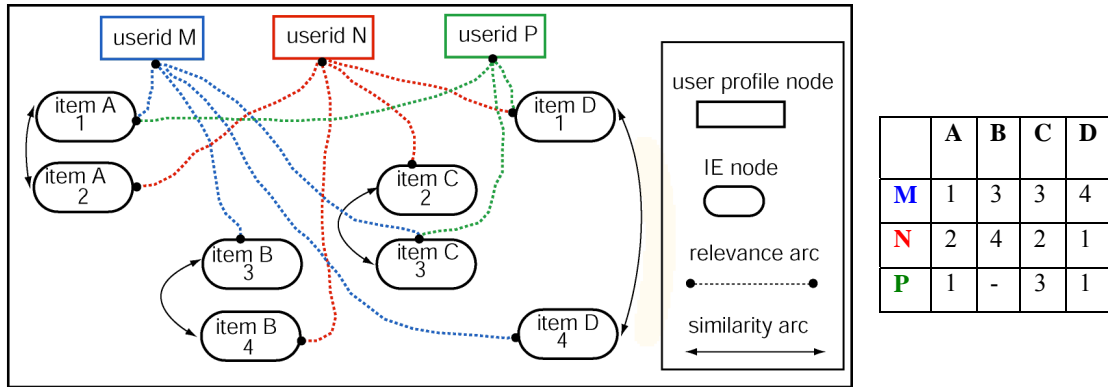


Figure 6.12: The architecture of a CRN using the user profiles, M, N and P and a set of ratings (1–5) for items A, B, C, D

Advantages of ACF CRNs:

CRNs are suited towards retrieval situations where there is missing attribute information. Case nodes are connected to IE nodes only on the basis of what attributes are available. Since cases share IEs, the memory required is significantly lower than that used by the flat memory structures typically used for ACF data. For instance, in the MovieLens dataset, an ACF CRN uses 6714 IE nodes to store 100,000 attribute values, and 16,912 IE nodes to store the 1,000,000 attribute value pairs of the largest publicly available MovieLens dataset. Intuitively, the maximum number of IEs required to represent a dataset in a CRN is the product of the number of attributes and the number of different values associated with each attribute. IE nodes in ACF are very simple data structures, themselves being little more than a holder for an attribute name and value, and an activation score.

As we shall see in the next section, CRNs alleviate the query time complexity for neighbour calculation by ensuring that only users who have IE nodes in common are considered, and by distributing the similarity calculation amongst shared IE nodes. Furthermore, the search through the ACF profile space is guaranteed to be complete (Lenz et al. 1998).

Another advantage is that new user profiles and ratings can be added without having to rebuild the memory structure which is necessary for ACF where the user ‘case base’ and the case profiles themselves are constantly growing.

Disadvantages

CRNs are suitable for numeric calculation where discrete values are used. In most ACF systems, a rating system is in place so that attribute values are discrete. For instance, Smart Radio ratings are chosen from 5 discrete values (1–5). The benefits of CRNs are lost when attributes use continuous values. In such a situation, a new IE must be created for each new value, and few cases will share common IEs. Thus the computational advantage of calculating similarity once for many connected cases is lost. Therefore, continuous values need to be discretised before being inserted into the

CRN. This problem arises for implicitly derived ratings. In Smart Radio, we round implicitly derived scores to the nearest 0.5. (i.e. an implicit score of 3.4 is rounded to 3.5).

CRN activation

ACF Neighbour Retrieval using a CRN has three stages:

1. *Initial Activation*: the target user profile is presented to the CRN and initial activation is determined for the CRN IE nodes that match the target features of the CRN.
2. *Similarity Propagation*: the second step involves incrementally propagating the activation through the connected IEs using the similarity function. In our ACF implementation the similarity function represents the contribution of an IE to the overall Pearson correlation. We explain this in more detail in the section below.
3. *Relevance Propagation*: the final step entails collecting the achieved activation in the associated case nodes. This is done using the relevance function which connects each case node to its constituent IE nodes. The relevance function performs the final Pearson calculation using the contributions of the activated IEs.

The result of the retrieval process is a set of cases ranked in order of decreasing activation. The activation constitutes a similarity connection between entities; it can be any measure that can be incrementally computed. In the context of our ACF implementation, the activation consists of a contribution of an IE to a Pearson correlation score between the target case and any cases connected to the IE via relevance arcs.

For complex measures like the Pearson coefficient, activation is an intermediary calculation in which several values are updated in each activated IE node. For instance, the Pearson correlation can be expressed using Equation 6.2. It is clear that the five subsections marked in the equation can be incrementally calculated, after which point a final calculation is carried out in which the square root of the denominator is computed. Using the Pearson metric, each activated IE contributes a partial calculation to the terms marked 1 to 5.

Equation 6.2

$$r = \frac{n(\sum XY) - (\sum X)(\sum Y)}{\sqrt{[n\sum X^2 - (\sum X)^2][n\sum Y^2 - (\sum Y)^2]}}$$

The final calculation is carried out by the *relevance function* which collects these values from each activated IE and performs the final calculation for each case node.

As we can see in Figure 6.12, an IE representing a single feature value pair is stored once irrespective of how often it occurs throughout the user–item matrix. Accordingly, the similarity calculation is only done once between a target IE and a corresponding IE in the CRN memory. This

calculation is then used by each case that is connected to that IE during the relevance propagation stage. We can use the simple CRN structure in Figure 6.12 to illustrate such a scenario. Let us take a sample target profile, T , and apply it to the CRN.

Table 6.2: A target profile for user, T

	A	B	C	D
T	2	4	3	1

The first stage involves initialising the activation of IEs corresponding to the target feature values. If a target feature is missing in the CRN it is created on the fly and inserted with the appropriate similarity links. This is illustrated by the IEs shaded in black. Activation is calculated for the target IE. The activation state of each node is shown in Table 6.3.

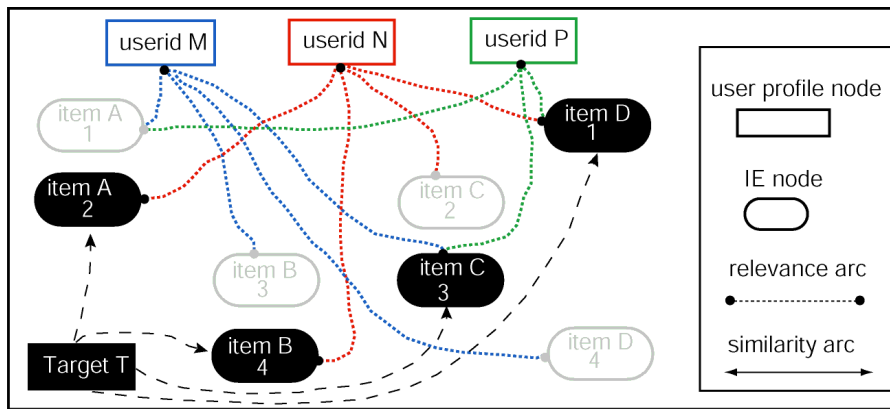


Figure 6.13: The first stage of activation. The nodes in black illustrate the activation of the target IEs.

In the second stage of activation, activation is calculated between the target IEs and any connected IEs. The activation in these is then set. This is illustrated by the IEs shaded in grey. For clarity each IE node has a connection with one other.

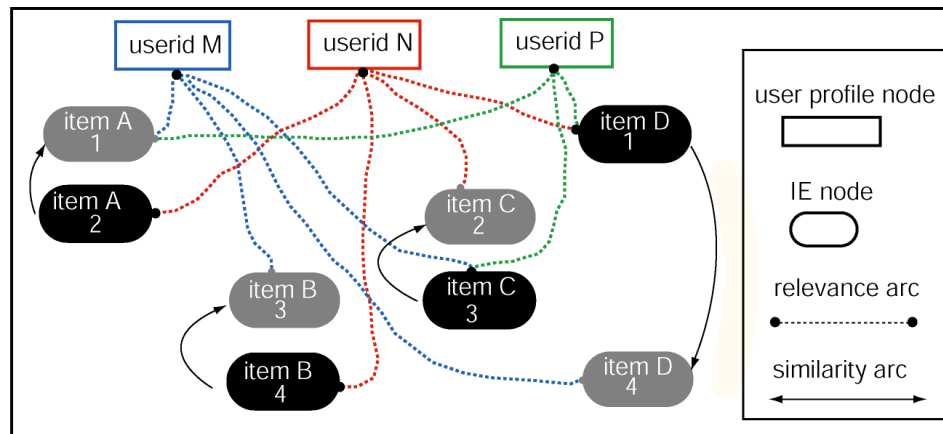


Figure 6.14: The second stage of activation. Activation is propagated through interconnected IEs.

Finally, the relevance function propagates the activation in each IE to connected case nodes and performs a final calculation.

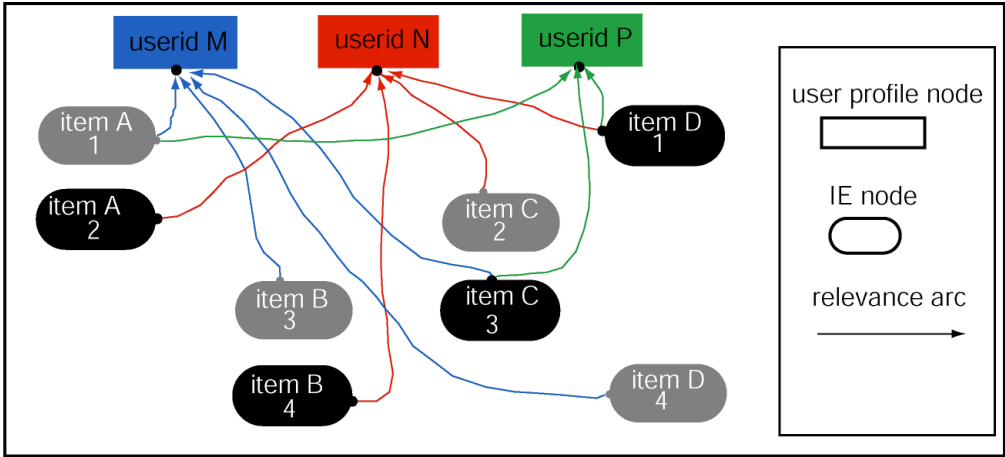


Figure 6.15: The final stage of CRN activation. The relevance function collects the activation in each connected, activated IE.

For the Pearson measure, each IE node has an activation state composed of 5 variables which, during relevance propagation, contribute to the labelled sections of Equation 6.2. Table 6.3 illustrates the activation states of each node after the first two stages of activation. The rows with a black background are nodes activated by the initial target activation (see Figure 6.13). The rows with the grey background show the states of IEs activated by the similarity propagation stage (see Figure 6.14). In this example the X attributes refer to the target attributes and the Y attributes refer to the attributes of the case being tested for similarity.

During activation, the node is passed the value for X (the target attribute), from which it can calculate XY, X², and update the X variable. These calculations are made once and then passed on to connected case nodes for final calculation of the Pearson coefficient. We further speed up query-time computation by pre-computing the value for Y² when the CRN is built. This value will be static for every correlation calculation. In Table 6.3 the labels marked in red indicate static values that can be pre-computed (Y²) or assigned (Y).

Table 6.3: The activation states of each node after the first two stages of activation

IE Node	XY	X	Y	X ²	Y ²
item A 1	2	2	1	4	1
item A 2	4	2	2	4	4
item B 3	12	2	3	16	9
item B 4	16	4	4	16	16
item C 2	6	3	2	9	4
item C 3	9	3	3	9	9
item D 1	1	1	1	1	1
item D 4	4	1	4	1	16

Thus the CRN similarity calculation is extremely efficient. Figure 6.16 illustrates the mean time per prediction using a flat search for the set of nearest neighbours vs. a search using a CRN. The dataset is the 100,000 MovieLens dataset. To start the experiment we removed 90% of the user profiles from the dataset, and using the *Leave_One_Out* technique made a prediction for each attribute-value in the dataset. The average prediction time is plotted on the Y axis. We then repeatedly replaced 10 % of the data and carried out the experiment at each increase of the dataset. As we can see, the increase in prediction time is linear with the CRN predicting at a significantly faster rate than the flat search.

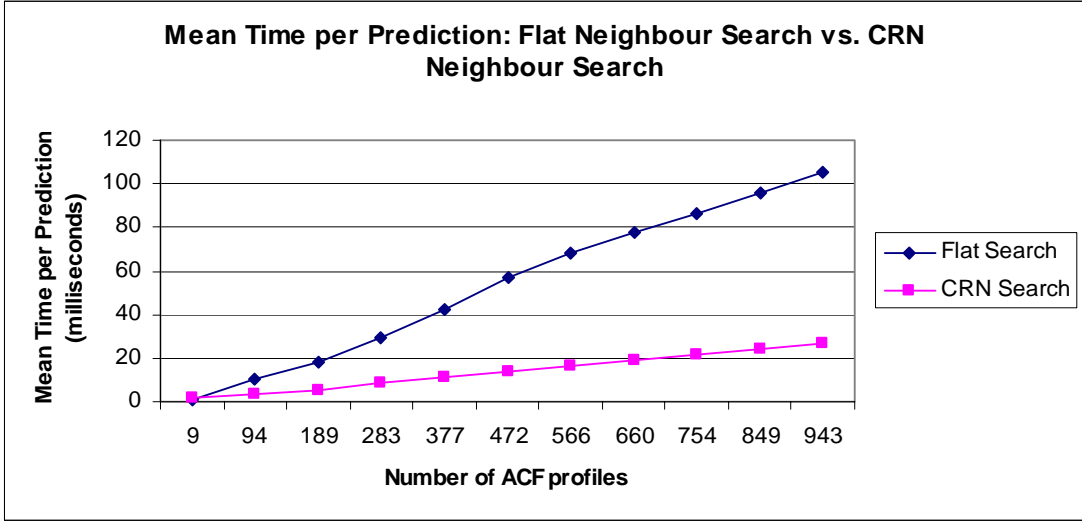


Figure 6.16: The mean time per prediction for a flat neighbour search vs. a CRN-based neighbour search

The improvement in prediction efficiency is due to the speed at which neighbourhoods can be calculated for each user. It has often been recognised in ACF literature that neighbourhood computation is a major bottleneck (Herlocker et al. 1999). Herlocker et al. (2000) suggest sampling user profiles to reduce the cost of this process. We would suggest that our technique offers the benefits of a complete search and increased speed. Furthermore, CRNs can be adapted for parallel processing (Lenz 1999), which is an advantage for systems with huge data repositories.

6.6.3 A Context Analysis Module

This module was developed to cater for recommendation strategies which require context specific information. It does this by maintaining a simple user profile that summarises the listener’s most recent listening interests. This is, in effect, a *sliding window* on the user’s listening history where the window contains the last n non-negatively rated tracks. The context analysis module produces a case-based profile, like the playlist case illustrated in Figure 5.5. In Smart Radio, we choose $n = 10$, which means that the output is generally a case-based representation of the last playlist played by the user. However, if there are negatively rated tracks in the playlist, these are not counted among

the n tracks. Instead the case representation will be based on an amalgam of the 10 most recent tracks that have not been rated negatively by the user (see Figure 6.17), which may contain tracks from the second most recent playlist.

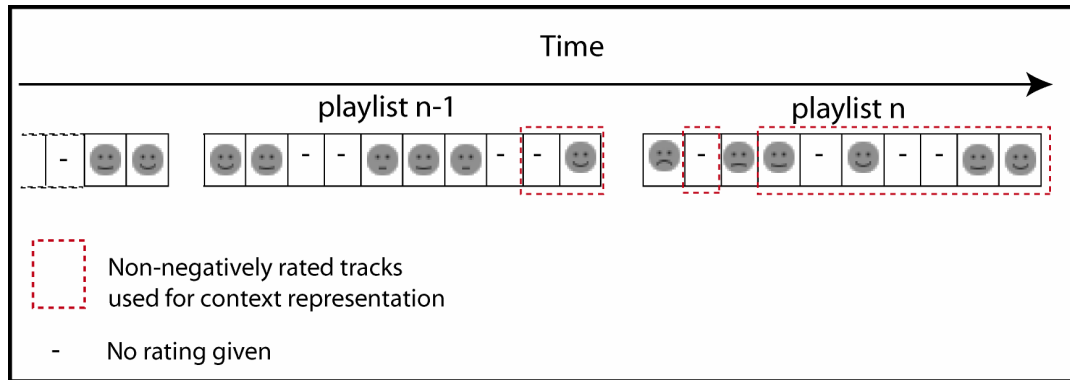


Figure 6.17: An example of the sliding window used to collect context data

The context analyser is called upon during the recommendation process in order to rank playlists according to their similarity to the sliding profile. 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 recommendation set.

6.6.4 Case-Based Reasoning Module

The CBR module in Smart Radio consists of a CRN in which playlists are represented as cases. Each case is 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 6.18, the CRN structure will only link those cases with features in common which ensures optimal retrieval while only traversing the relevant portions of case memory (Lenz 1999).

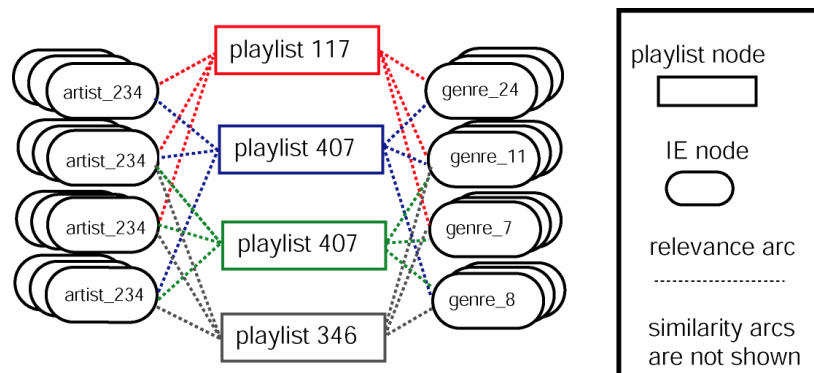


Figure 6.18: A schema of the playlists indexed using a CRN

The CBR module is used for similarity-based ranking of ACF recommendations in which it receives a target case as input from the context analysis module (see Figure 5.10). Since only a portion of the case base (a set of recommendations) needs to be considered for each ranking operation, the module can receive *priming* input directly from the ACF module or from the database connectivity component. This input *turns on* the portion of the CRN that should be used for retrieval. The Playlist CRN is also used to return similar playlists to the user based on an example playlist selected by the user. This is discussed further in the next section.

6.6.5 Recommender Services

Our recommendation engine operates as a multi-threaded server using RMI (Remote Method Invocation) as its communication protocol. Within the server, application logic and connection logic are separated. The server has two modes of operation, asynchronous and synchronous. When a synchronous request is received, an RMI connection thread contacts an application thread. If the application thread does not respond within a predetermined time, the RMI connection thread returns a time-out message to the client. With asynchronous requests, each request is queued and is serviced by the next available application thread. Once the request is queued for processing the connection thread returns a message confirming that the request has been scheduled.

In the case of Smart Radio, the client is either the servlet engine, or the database manager. Smart Radio users do not have direct access to the recommendation server. In order to manage the request load we use asynchronous requests for scheduled data management tasks, and recommendation updates. Such requests, although time sensitive, do not require a timely response.

Asynchronous requests

The flowchart in Figure 6.19 illustrates the data management process and the recommendation update process. The data update process, which we described in Section 6.4 is represented on the left of the chart in dark grey. The second column from the left in the flow chart represents the steps in the process that directly affect the recommendation server. As we can see there, the internal data model of the recommendation server is updated for selected user profiles, after the ACF tables are updated. Once this has been completed the recommendation notifies the *remote user object* of the date and time of the data update. The *remote user object* is an object, representing users who are currently on-line, whose methods can be remotely invoked by the recommendation server, the servlet engine or the scheduling component. It provides a communication interface between these components.

The second process represented in the flow chart in Figure 6.19 is the recommendation display/update process, shown in light grey to the right of the chart. The recommendation update process uses a combination of lazy invocation and scheduling. The basic idea is that a recommendation update is calculated for each on-line user periodically, but only if he/she has

submitted feedback in the interval. We define the *recommendation_period_threshold* as the time interval after which a set of recommendations for an on-line user is recalculated, if the user has submitted data since the last recommendation update.

Thus, when a user requests the recommendations page, the web server firstly passes the request to the servlet engine which invokes the recommendation servlet. The servlet then checks whether new recommendations are required to be generated by the recommendation server. It does this by checking two pieces of information on the *remote user object*. The *rec_date* property gives the time of the last recommendation update for the user, and the *recent_data_date* property gives the time of the last feedback from this user to have been recompiled into the server's internal data model. The servlet issues an asynchronous request to the recommendation server, if the two following conditions are met

1. If the time elapsed since the *rec_date* is greater than the *recommendation_period_threshold*.
2. If the *recent_data_date* is greater than the *rec_date*.

We use this combination of scheduled and lazy invocation for the recommendation process for two reasons. Firstly, as each recommendation request is relatively computationally expensive, we carry it out only if necessary. For example, even if the threshold period for generating recommendations has been exceeded, a recommendation update will not be requested until the user has submitted feedback and this data has then been compiled into the internal data model of the recommendation server. This ensures a level of consistency in the user's interaction with the system – users will expect their recommendations to change only after they have submitted feedback.

Using a recommendation strategy that reacts immediately to user feedback leads to the problems we encountered in the first version of Smart Radio where users would enter bogus ratings in order to see what the recommendation engine would produce. We introduced the *recommendation_period_threshold* to delay the effect that new ratings have on recommendations, which should go some way towards frustrating the intentions of such users. Currently the *recommendation_period_threshold* is set at 20 minutes.

Deployment Flowchart: Asynchronous Data update and Asynchronous recommendation update

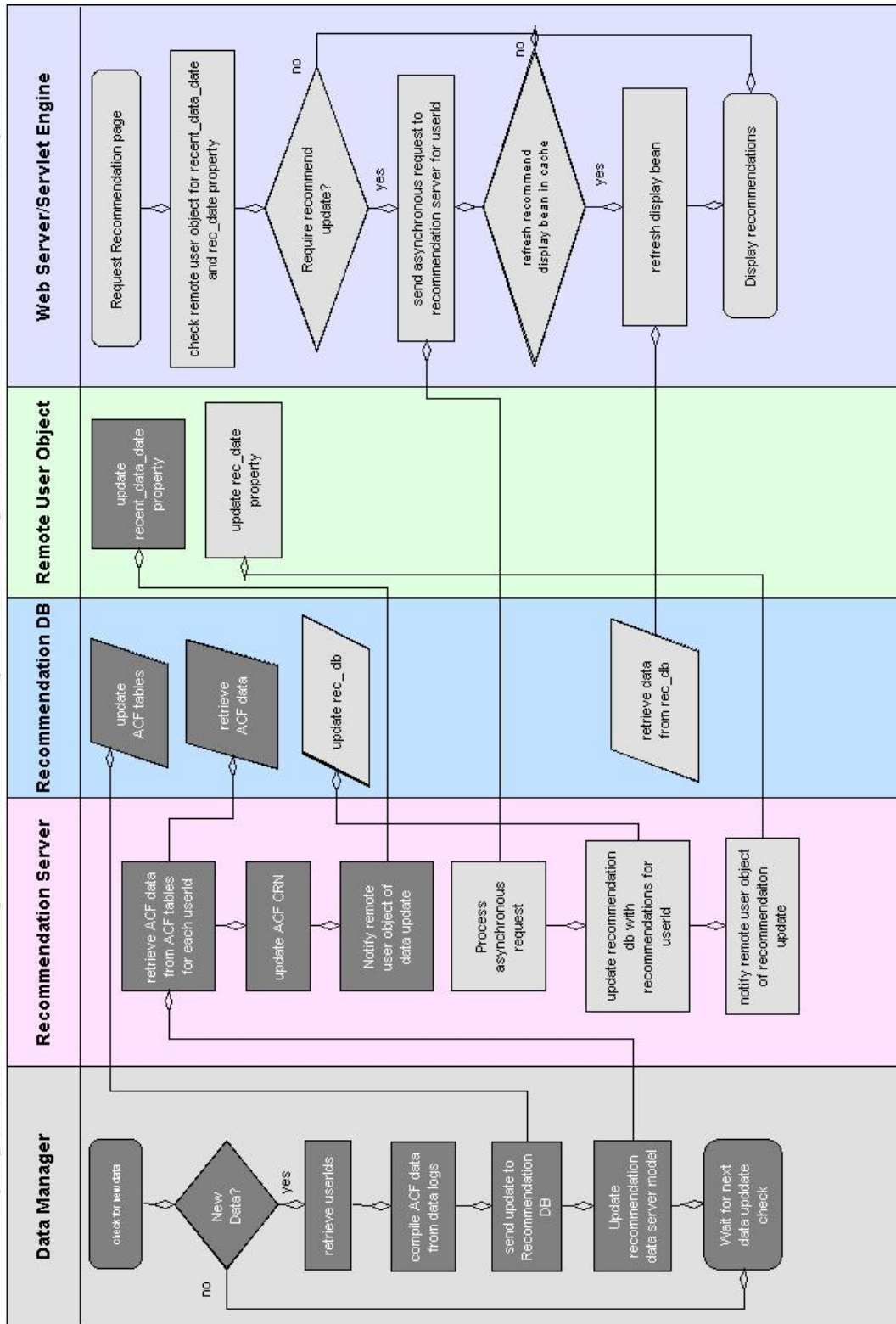


Figure 6.19: A flow chart demonstrating two asynchronous process flows

Synchronous requests

However, there are events in the system for which we require an immediate update to the recommendations for a particular user, in which case we use synchronous processing. Because of

their timing constraints synchronous requests have priority in the recommendation server. They are placed in a separate queue to the asynchronous requests. When application threads are free, this queue is serviced before the asynchronous request queue.

Bootstrapping new users

The first event for which we call upon synchronous request processing is the bootstrapping of new users. In Section 6.5.3 we describe how new users are asked to rate several artists. It is important that we can promptly provide these users with recommendations after they submit their ratings. In Figure 6.10 we provide evidence that the recommendation server should quickly be able to calculate recommendations using the user–artist data matrix. However, as it is possible for a synchronous request to time-out, our fall back strategy is to present the top ten playlists in the system at the time along with an apology. Synchronous requests that have timed-out will still be processed by the next available application thread, so that a set of new recommendations should be presented when the user refreshes the recommendation page.

Context ranking

The second event for which we employ synchronous requests is the calculation of user-context and the ranking of the recommendation set according to that context. One of the key recommendation strategies employed by Smart Radio is the boosting of recommendations according to the user’s perceived interests at the time. When a user plays a playlist we immediately re-rank his/her recommendations to reflect similarity to the last ten items that the user has not rated negatively. To re-rank users’ recommendations after a playlist event we employ a synchronous approach which is illustrated in Figure 6.20. After the user selects a playlist to play, the servlet engine generates a synchronous ‘context’ request and sends this to the recommendation server. The recommendation server calls the context analyser module which builds a case base representation of the user context. The playlist case base is then primed with the user’s ACF recommendations. The context case is presented as a target case to the case base and the top n ranked cases are determined to be the context-based recommendations that will be presented to the user. These cases are removed from the set of ACF recommendations and pushed to the top of the recommendation set that is presented to the user.

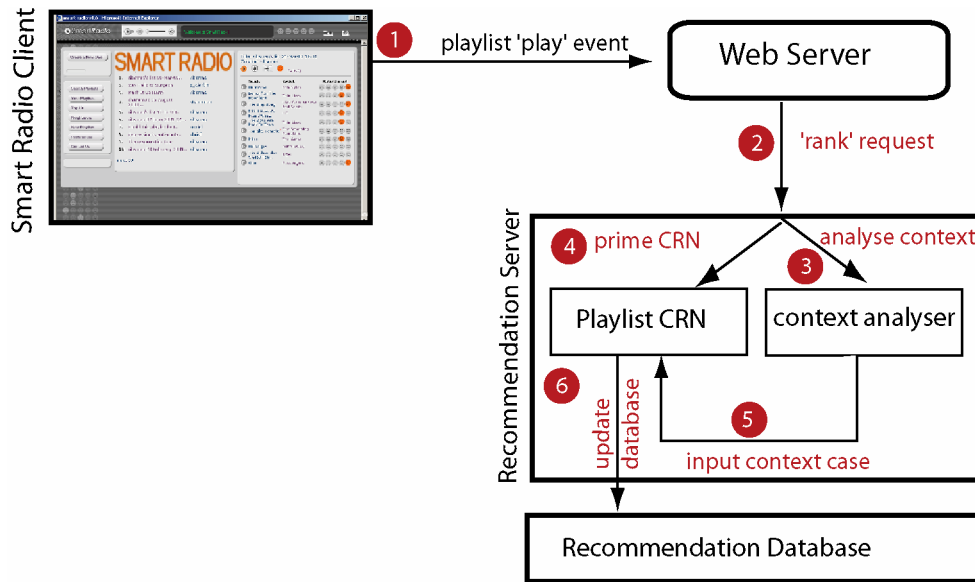


Figure 6.20: A synchronous request updates the recommendation database in response to a playlist being played

Users quickly learn that the recommendation set follows the trends in their listening preferences. In order to avoid any confusion about why the recommendation set has changed, we provide access to an explanation for each recommended playlist. As shown in Figure 6.21 there are two explanations. The first is a context-based explanation, and the second is an ACF-based explanation. We also make reference to the user’s novelty preference which influences the ACF ranking. Swearingen and Sinha (2001) and Herlocker et al. (2000) have found that user trust in the recommendation strategy is bolstered by revealing the reasoning behind each recommendation.

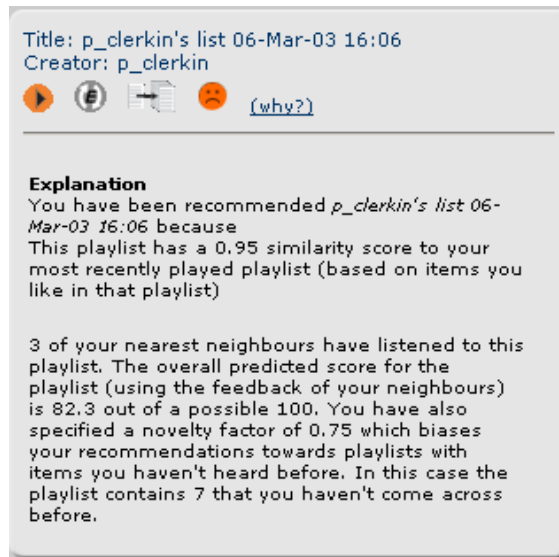

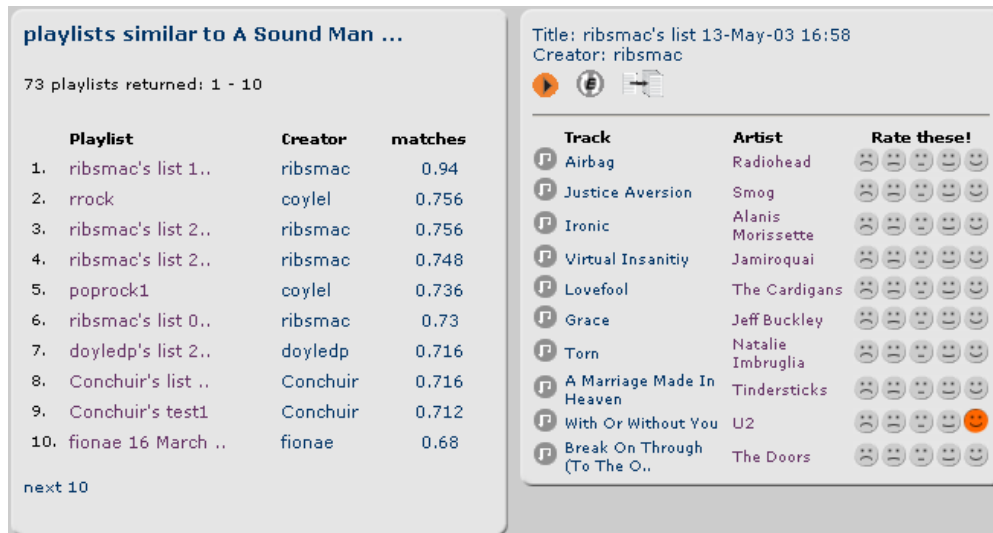


Figure 6.21: Each recommended playlist is accompanied by an explanation

Query by playlist

By clicking on the query by playlist  icon Smart Radio users can retrieve a set of playlists similar to the one they are currently viewing (see Figure 6.22). This facility allows listeners to zoom in on portions of the playlist case base of particular interest to them. When processing such a request the servlet engine dispatches a synchronous request to the recommendation engine, and waits for a response. When a response is received, the servlet engine invokes the JSP engine which dynamically builds a web page around a list of playlists received from the recommendation server.



The screenshot displays two panels. The left panel, titled "playlists similar to A Sound Man ...", shows a list of 10 playlists with their creators and match scores. The right panel, titled "Title: ribsmaac's list 13-May-03 16:58", shows a list of tracks with their artists and a "Rate these!" interface with smiley faces.

Playlist	Creator	matches
1. ribsmaac's list 1..	ribsmaac	0.94
2. rrock	coylel	0.756
3. ribsmaac's list 2..	ribsmaac	0.756
4. ribsmaac's list 2..	ribsmaac	0.748
5. poprock1	coylel	0.736
6. ribsmaac's list 0..	ribsmaac	0.73
7. doyledp's list 2..	doyledp	0.716
8. Conchuir's list ..	Conchuir	0.716
9. Conchuir's test1	Conchuir	0.712
10. fionae 16 March ..	fionae	0.68

Track	Artist	Rate these!
Airbag	Radiohead	☹ ☹ ☹ ☹ ☹
Justice Aversion	Smog	☹ ☹ ☹ ☹ ☹
Ironic	Alanis Morissette	☹ ☹ ☹ ☹ ☹
Virtual Insanitiy	Jamiroquai	☹ ☹ ☹ ☹ ☹
Lovefool	The Cardigans	☹ ☹ ☹ ☹ ☹
Grace	Jeff Buckley	☹ ☹ ☹ ☹ ☹
Torn	Natalie Imbruglia	☹ ☹ ☹ ☹ ☹
A Marriage Made In Heaven	Tindersticks	☹ ☹ ☹ ☹ ☹
With Or Without You	U2	☹ ☹ ☹ ☹ ☹
Break On Through (To The O..	The Doors	☹ ☹ ☹ ☹ ☹

Figure 6.22: The *Query by Playlist* facility allows users to request playlists *similar* to the current playlist

6.7 Conclusion

In this chapter we concentrated on the data architecture of the Smart Radio system. This architecture is concerned with the collection and manipulation of data to provide timely recommendations to our users. We use a combination of synchronous and asynchronous services to achieve that end. Our Data Manager converts log data into ACF data, which is then used to update the data model in the Recommendation Server. The Data Manager can derive implicit scores for track and artist data. We show that using implicitly derived artist scores helps us during the bootstrap phase, allowing the recommender to produce better predictions when a reduced amount of data is available. We demonstrate that Smart Radio is reliant upon the industry of a few playlist compilers whose names become well known throughout the user community, and we suggest that the role of *community leader* is the primary incentive for such users.

Since ACF is a lazy, data intensive recommendation strategy, we look at reducing the memory overhead and query-time computational expense. We suggest that the ACF technique is similar to a technique in CBR called case completion. We show how a memory structure used in

CBR can be applied with ACF. A Case Retrieval Net reduces memory overhead by storing each feature-value pair once as an information entity. Thus several cases may share the same information entity. Retrieval time is speeded up because local similarity calculations are performed once for each IE and distributed to case nodes that share the IE. As an example, we show how the calculation of the Pearson coefficient can be distributed among the IEs of the CRN during retrieval time.

Chapter 7: Evaluating Smart Radio

Increasingly, there has been a demand for objective evaluation criteria for automated recommendation 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. There are many aspects of a recommender system we could analyse (Weibelzahl 2001). For instance, the ease with which it can be used is certainly an important factor for success. This has much to do with HCI factors such as presentation and the interaction model employed. The latter has sometimes been addressed by comparing the dialogue complexity of competing algorithms and in this case the system with the shorter dialogue is judged to be the winner (Doyle & Cunningham 2000). The most common evaluation approaches are performed off-line using techniques from machine learning and information retrieval such as cross validation and measures of recall/precision.

We suggest that two types of evaluation are required for a recommender system: an *off-line* and an *on-line* approach. An off-line approach can demonstrate that the algorithms are performing within limits recognised in the research literature. However, an off-line approach is limited in the types of recommendation strategy it can evaluate. For instance, it is not suitable for comparing strategies that provide different types of ranking such as ACF and context-boosted ACF. Nor can it demonstrate whether one strategy actually performed better than another when deployed in a live application with real users.

In order to test this we use an *on-line* evaluation. The basic idea behind this type of evaluation is 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 the relative degree of success of each strategy according to how the user utilises the recommendations of either system.

In Section 7.1 of this chapter we describe an off-line analysis of our ACF strategy. This process will tell us a lot about the data underlying our recommendation system – its ability to build neighbourhoods of users, the prediction error rates and coverage. It will highlight weaknesses in our current recommendation strategy, and help us understand how we might overcome these problems.

In Section 7.4 we present a novel on-line evaluation in which a pure ACF strategy and a context-boosted ACF strategy are concurrently deployed. Our goal is to measure whether our users find context-boosted ACF an improvement over standard ACF. Since the context-boosting technique imposes a ranking based on the user's current listening preferences, the on-line evaluation tests whether the user was inclined to make use of the recommendations presented to him/her. Both strategies *simultaneously* compete to give recommendations to the *same* user at the

same time. Our goal is to obtain a relative measure of user satisfaction with respect to both recommendation strategies. We examine how we might maintain user trust during the evaluation period, and how we might offer each recommendation engine equal opportunity for success.

7.1 Off-line Evaluation

Off-line evaluation uses existing datasets such as the publicly available EachMovie dataset or data gathered during the operation of a recommender prototype (Hill et al. 1995). In off-line evaluation, recommendation can be viewed as information retrieval, i.e. the selection of the subset of assets that are relevant to the user. From this perspective the metrics for evaluation are the well-known measures of precision and recall (Baeza-Yates & Ribeiro-Neto 1999). Precision is the proportion of retrieved assets that are relevant and recall is the proportion of relevant assets actually retrieved. However, a problem lies in defining *a priori* what the set of relevant assets is in a recommender system. Basu and Hirsh (1999), in evaluating their recommender system, defined the upper-quartile of the movies rated by the user as the relevant set. So the precision is the percentage of upper-quartile movies in the set returned. The alternative to viewing recommendation as an information retrieval problem is to view it as a classification or regression problem. In a situation where users have rated assets, the recommendation problem may be cast as the prediction of these ratings – a regression problem. Alternatively, it may be viewed as a classification problem – the classification being whether an asset will be liked or disliked. The measure of accuracy may be: 0/1 error for classification, mean absolute error (MAE) or root-mean-square (RMS) error for regression or a measure of the correlation of predicted ratings with actual ratings (e.g. using the Pearson correlation coefficient). This allows for the type of evaluation that is common in machine learning where the data is partitioned into training and test data – using the training data to produce predictions for the test data. These estimates may be improved by using *k*-fold cross-validation or by using a *Leave-One-Out* evaluation, i.e. a rating is predicted by using all the data except the rating itself (Breese et al. 1998).

Our off-line evaluation draws upon existing work for the evaluation of ACF systems (Herlocker et al. 1999, Breese et al. 1998, Miller et al. 1997, Basu et al. 1998, Billsus & Pazzani 1998). We have chosen not to evaluate precision and recall, and simply defined another measure called *coverage* which is the percentage of target items the algorithm was able to make a prediction for. This is a more suitable measure for our domain, since we may need to make a prediction on each track in each playlist in our candidate set.

7.1.1 Evaluation Set-up: Leave-One-Out

Following the example of Herlocker et al. (1999) and Breese et al. (1998) we use MAE as a measure of the accuracy of our implementation. The testing technique we use is a variant of *Leave-one-out* (Moore & Lee 1994) cross-validation adapted for ACF. We calculate the MAE for each user instance in the dataset and average the results to obtain an overall MAE.

To calculate the MAE for a single user (the test user), we withhold the user’s data from the dataset (*leave-one-out*). Then we iterate through the test instance rating data, withholding one item at a time (the test item). We predict a value for the test-item using Equation 4.7, where correlations with instances in the training data are based on all ratings in the test user *minus* the test item. Thus, for each test item a new neighbourhood is assembled based on the remaining data in the test user instance. This evaluation is a more thorough version of that carried out by the Herlocker et al. (1999) where a ten fold cross validation was used, and where 5 random items were chosen from each test user. We chose this methodology because the Smart Radio dataset is considerably smaller than the MovieLens dataset. Accordingly the *leave-one-out* technique allows us to keep as much as possible of the data in the training portion of the set.

7.1.2 Baseline

As a baseline by which we can judge the performance of our system on the different dataset configurations, we use the mean absolute error per user obtained when predictions are made using the *average* score for each item across all users *except* the test user. Since ACF uses a weighted average across a subset of the user base to make predictions, if it cannot do better than the baseline technique there is insufficient data available to make personalised predictions. A similar strategy was used by Shardanand and Maes (1995) to evaluate an ACF algorithm in the Ringo system.

7.1.3 Comparison with MovieLens

In order to test our ACF engine we used the MovieLens dataset on which there have been previously published results (Herlocker et al. 1999, Sarwar et al. 2000b). The goal is to ensure that our implementation of ACF can duplicate the benchmark results obtained on this dataset.

This evaluation demonstrated that our implementation could reproduce the same results obtained by Herlocker et al. and Sarwar et al. Setting the maximum neighbour size, *k*, equal to 50 in both implementations achieves exactly the same MAE (0.7678).

Table 7.1: A summary of the data configuration used in our evaluation (and that of Herlocker et al. 1999) of the MovieLens dataset

Dataset	Details
1. MovieLens	<p>Description: movie data capturing explicit voting preferences over a selection of 1682 movies</p> <p>Number of user instances: 943</p> <p>Dimensions: 1682</p> <p>Total rated items: 100,000</p> <p>Average number of items per user: 106.04</p> <p>Constraints: users have explicitly rated at least 15 items.</p> <p>Sparsity: 0.9369</p>

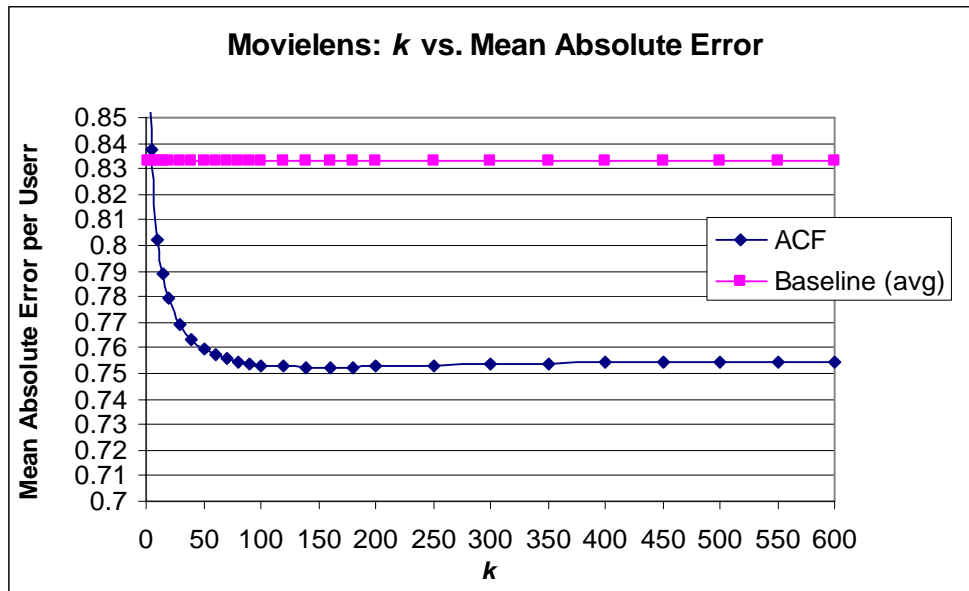


Figure 7.1: A plot of neighbourhood size vs. Mean Absolute Error

The graph in Figure 7.1 demonstrates a rapid decrease in MAE as neighbourhood size increases. This is followed by a gradual increase as more neighbours are added. The increase in MAE can be attributed to noise added by neighbours with low correlation scores (Herlocker et al. 1999). For comparison purposes we include the MAE achieved using the prediction score calculated using the baseline strategy, which performs quite well on this dataset.

We also include a measure of *prediction correlation* for comparison with the Smart Radio datasets. This score indicates how well the prediction task can follow the rating patterns of users in the dataset. We use the Pearson correlation coefficient, which was described in Chapter 4. As the correlation coefficient is scale independent, it is sensitive to the different means in each user’s rating set. Thus we should expect to see a better correlation score for the ACF strategy than for the baseline technique.

For this experiment we set $k = 100$, which is midway between 80 and 120, the optimal range for k suggested by Sarwar et al. (2000b) for the MovieLens dataset. Our experiments would seem to verify this observation (see Figure 7.1). Table 7.2 shows the results of the experiment. As expected, the correlation coefficient is higher than for the baseline technique, though not by a great margin.

Table 7.2: The results achieved using the ACF at $k = 100$ to make predictions on the MovieLens dataset

Dataset: MovieLens	ACF, $k = 100$	Baseline Evaluation Results
MAE per user	0.7564	0.8328
Prediction Correlation	0.556	0.4212
Coverage %	99.04	99.859

7.1.4 Smart Radio Dataset Configurations

Due to the constraints on listenership imposed by IMRO described in Chapter 2, the Smart Radio dataset is considerably smaller and sparser than the MovieLens dataset. This lack of data will detract from the performance of the ACF algorithm. However, we use the baseline predictive strategy as a performance benchmark for each test set.

We evaluate four dataset configurations, which are discussed in the next section and summarised in Table 7.3.

1. Explicit track data (`track_explicit`).
2. Explicit track data augmented by implicitly derived track data (`track_explicit_and_implicit`). The algorithm for deriving the implicit data is described in Chapter 6.
3. Dimensionally reduced data (`artist_implicit`).
4. Dimensionally reduced data augmented by explicit artist data (`artist_implicit_and_explicit`).

The key dataset is `track_explicit` since the others are augmented (`track_explicit_implicit`) or dimensionally reduced versions (`artist_implicit` and `artist_implicit_explicit`) of this dataset.

7.1.5 Discussion on Dataset Configurations

Dataset 1: Explicit track data

The data collected in this dataset form the basis of all our experiments. It consists of 71 Smart Radio listeners who have explicitly rated 5 or more music tracks. Although we have collected data for more than 145 listeners, many listeners have not given us explicit ratings at all. Since we have no mean around which we can infer ratings for these users we do not use these ratings when we are making ACF-based predictions for other users. Our experiments suggest that ratings that are completely implicitly derived ratings are the source of considerable noise in the full dataset. We also do not make use of users who have rated less than 5 items explicitly. Other experiments demonstrate that users who have contributed less than 5 ratings have a much higher mean error than users who have rated 5 or more music items. The same type of data filtering has been undertaken in the MovieLens dataset where users who have rated less than 15 movie items are excluded from contributing to the ACF prediction process.

This dataset has a higher dimensionality than the MovieLens dataset with 4131 items for which each user can have a rating. Thus, as well as having less user instances, it has a much higher sparsity level (0.9734) than the MovieLens dataset.

Dataset 2: Explicit track data augmented by implicitly derived data

This dataset consists of the explicit track data, described above, and an additional set of ratings derived according to the implicit score algorithm described in Chapter 6. This algorithm uses the mean and standard deviation of each user's explicit rating set to construct implicit scores for items the user has listened to but not rated. The implicitly derived dataset contributes 6735 ratings to the existing 71 users in the `track_explicit` dataset, lowering the data sparsity to 0.9504. When we evaluate this dataset, we make predictions for the explicitly rated items only, so as not to predict any bias in the implicit data algorithm. Thus our evaluation should demonstrate whether the implicit data improves upon the predictions made using the explicit dataset alone.

Dataset 3: Implicit artist data

In Section 6.5.3 we described the algorithm the data manager uses to reduce the dimensions of the user-item matrix. In the context of Smart Radio, using the user-artist matrix allows us to more effectively bootstrap users into the system. As we demonstrated in Section 6.5.4, we achieve better prediction accuracy when we elicit 7 *artist* ratings rather than 7 *track* ratings from the new user.

Applying the dimensionality reduction transform (see Equation 6.1) to the `track_explicit` dataset (dataset 1), we obtain a dataset with the same number of users but with the item dimensions reduced from 4131, the number of music items, to 333, the number of artists represented in the dataset. This considerably reduces the sparsity of the dataset to 0.8632. When we are evaluating this dataset (dataset 3) we use neighbours correlated using this data to make predictions for the explicitly rated items in dataset 1. We use a modified form of the `Leave_One_Out` evaluation to test this dataset. For every test instance, we recalculate the mean score for the associated artist feature from the explicit track scores *minus* the test item score. Thus for each test item to be predicted a new artist feature score is re-calculated.

Dataset 4: Implicit artist data and explicit artist data

The bootstrap facility whereby new users could explicitly rate a set of artists was introduced to the Smart Radio system approximately 11 months prior to the time of evaluation. At the same time we extended the facility to allow existing users to rate artists if they wanted to. Thus our explicit artist dataset is relatively small. Of the 71 users in dataset 3, 39 have *explicitly* rated artists producing a total of 961 artist ratings. We add these to dataset 3 to produce dataset 4. We overwrite the implicitly calculated score for a user with the explicit score where there is a conflict. This leads to 768 implicitly calculated scores being overwritten with explicit scores, and the addition of 193 new explicit scores. Thus dataset 4 is marginally less sparse than dataset 3. Our interest lies in examining whether the addition of explicit scores improves upon the performance obtained by dataset 3. As with dataset 3, we use the modified version of `Leave_One_Out` to test whether we can predict the explicit scores from test set 1.

Table 7.3: This table summarises the data configurations used in our evaluation

Dataset	Details
1	<p>Dataset: <code>track_explicit</code> Description: explicitly rated music items Number of user instances: 71 Dimensions: 4131 Total rated items: 7808 Average number of items per user: 109.97 Constraints: users have explicitly rated at least five items. Sparsity: 0.9734</p>
	<p>Evaluation: <code>Leave_One_Out</code> Test Set: explicitly rated items sampled using the <i>Leave-One-Out</i> Method. 7808 items from 71 users Training Set: explicit data other than the test item.</p>
2	<p>Dataset: <code>track_explicit_implicit</code> Description: 7808 explicitly rated music items plus 6735 implicitly rated music items. Number of user instances: 71 Dimensions: 4131 Total rated items: 14543 Average number of items per user: 204.83 Constraints: users have explicitly rated at least five items. Sparsity: 0.9504</p>
	<p>Evaluation: <code>Leave_One_Out</code> Test Set: explicitly rated items sampled using the <i>Leave-One-Out</i> Method. 7808 items from 71 users Training Set: explicitly rated items other than the test item.</p>
3	<p>Dataset: <code>artist_implicit</code> Description: 3116 implicitly rated artists. The scores are derived from the explicit music item scores in the <code>track_explicit</code> dataset. Number of user instances: 71 Dimensions: 333 Total rated items: 3116 Average number of items per user: 43.887 Constraints: users have explicitly rated at least five music items from <code>track_explicit</code> dataset. Sparsity: 0.8682</p>
	<p>Evaluation: <code>Leave_One_Out</code> (modified) Test Set: explicitly rated items from the <code>track_explicit</code> dataset sampled using the <i>Leave-One-Out</i> Method. 7808 items from 71 users. Training Set: explicitly rated music items other than the test item (<code>artist_implicit</code> scores derived from this data).</p>
4	<p>Dataset: <code>artist_implicit_explicit</code> Description: The 3116 implicitly rated artists from the <code>artist_implicit</code> dataset plus 961 explicit artist ratings. 768 implicit ratings were overwritten with the explicit scores. 193 new explicit scores were added. Number of user instances: 71 Dimensions: 333 Total rated items: 3309 Average number of items per user: 46.6 Constraints: users have explicitly rated at least five music items from <code>track_explicit</code> dataset. Sparsity: 0.86</p>
	<p>Evaluation: <code>Leave_One_Out</code> (modified) Test Set: explicitly rated items from the <code>track_explicit</code> dataset sampled using the <i>Leave-One-Out</i> Method. 7808 items from 71 users. Training Set: explicitly rated music items other than the test item, plus explicit artist ratings (<code>artist_implicit_explicit</code> scores derived from this data)</p>

7.1.6 Evaluation

Our off-line evaluation has the following objectives:

1. To demonstrate that ACF performs better than the baseline method.
2. To demonstrate that ACF performs according to previously observed norms.
3. To evaluate which dataset configuration performs best. We define some performance measures in the following paragraph.

For each dataset configuration we demonstrate the following:

1. *Mean neighbourhood size* vs. maximum neighbourhood size, k . This will demonstrate the ability of the dataset to produce neighbourhoods of users. This is directly related to the sparsity of the dataset. The sparser the dataset, the less likely there will be overlap between users. Intuitively, neighbourhood size will also be affected by the number of users represented in the dataset.
2. *Mean absolute error* (MAE) per user vs. maximum neighbourhood size, k . For each dataset we are interested in finding the value of k that will provide us with the lowest mean absolute error and maximum coverage.
3. *Prediction correlation* with respect to the test dataset.
4. *Coverage* vs. maximum neighbourhood size, k . We define coverage as the percentage of test items we are able to make predictions for.

Additionally, for dataset configurations 3 and 4, in which we have considerably reduced the dimensionality of the datasets, we are interested in the comparative improvement in the rate at which predictions can be made.

Prediction mechanism:

The prediction mechanism is ACF using a *significance threshold* and an *overlap threshold*. As described in Chapter 4, a *significance threshold* allows us to adjust the correlation score of users who have fewer items in common than is specified by the threshold. An *overlap threshold* disallows correlation scores that are based on less than the number of items specified by the threshold.

Each dataset configuration was tested using different significance thresholds. For `track_` configurations (configurations 1 and 2) a *significance threshold* of 40 was found to give the best MAE at $k = 70$. For `artist_` configurations (configurations 3 and 4) a threshold of 15 was found to give the lowest MAE at $k = 70$.

Likewise, *overlap thresholds* of 3 and 2 were found to be optimal at $k = 70$ for the `track_` configurations and the `artist_` configurations respectively. Intuitively, the *overlap threshold* has to be lowered for these datasets because dimensionality reduction will decrease the number of items in common.

7.1.7 Baseline Evaluation

The results of the Baseline Evaluation are given in Table 7.5. Unlike the evaluations to follow, these results are constant. Therefore we do not provide a separate graph for this evaluation.

Table 7.4: A summary of the evaluation set up for the baseline evaluation

Base Line Evaluation: Average explicit scores
Dataset Configuration: <code>track_explicit</code>
Test set: explicit data sampled using the Leave-One-Out Method. 7808 items from 71 users
Training set: explicit data other than the test instance.

Table 7.5: A summary of the results for the baseline evaluation

Baseline Evaluation Results	
MAE per user	1.0354
Prediction Correlation	0.182
Coverage %	94.595

This evaluation provides a basis for measuring whether our dataset can provide personalised recommendations using ACF. The MAE per user is 1.0354. This is the mean error over a 1–5 rating scale which is the same scale used by the MovieLens dataset. This figure is much higher than that achieved using the MovieLens data (0.8328). The baseline algorithm also produces a much lower prediction correlation with this dataset than with the MovieLens data. We attribute these affects to the sparsity of the dataset, which causes single item scores to have a significant influence on the calculation of the mean. We would expect the same effect to be observed in the ACF calculation, but to a lesser extent since the prediction algorithm uses a weighted mean. The coverage figure in Table 7.5 represents the upper bound for the percentage of items that can be predicted in this dataset. This score cannot reach 100% because there are 422 items out of a total 7808 (5.4%) that have been rated by one person alone. Thus the baseline technique and ACF techniques cannot be used to produce a prediction for these items.

7.1.8 ACF Evaluation

Test 1: k vs. mean neighbourhood size

The graph in Figure 7.2 plots the maximum neighbour size allowed, k , against the average neighbourhood size achieved for all four datasets. Examining the `track_explicit` dataset first, we see that the average neighbourhood size converges at 29.54 for $k = 70$, the maximum value for k . This relatively low figure can be explained by the sparsity of the data and the paucity of users. Even though we increase k , correlations between many users are not possible because they have not rated any items in common. However, the `track_explicit_implicit` data, which has the

same number of users but is less sparse, produces larger mean neighbourhood sizes at each increment of k .

The `artist_implicit` and `artist_implicit_explicit` datasets each have the same user set but are again less sparse. We observe that the mean neighbourhood size for each increment of k is accordingly larger. The least sparse dataset, `artist_implicit_explicit`, is able to build larger neighbourhoods at a faster rate than the other datasets. While the ability to build neighbourhoods in ACF is crucial to prediction performance, since each prediction is made using the accumulated data from the neighbourhood, the quality of the prediction will depend on the underlying rating data. If this data is noisy, it will lead to inaccurate correlations and poor neighbourhoods for predictive purposes. This is particularly pertinent to implicitly derived data, which by its nature contains noise. If the assumptions that underlie the creation of the implicit data are wrong, then noisy implicit scores will be created. These scores will lead to poor correlations and poor neighbourhoods for predictive purposes. Therefore, in the case of the `track_explicit_implicit` and `artist_implicit_explicit` datasets we will be keen to observe whether the addition of implicit data increases MAE and/or decreases prediction correlation at the expense of increased coverage due to increased neighbourhood size.

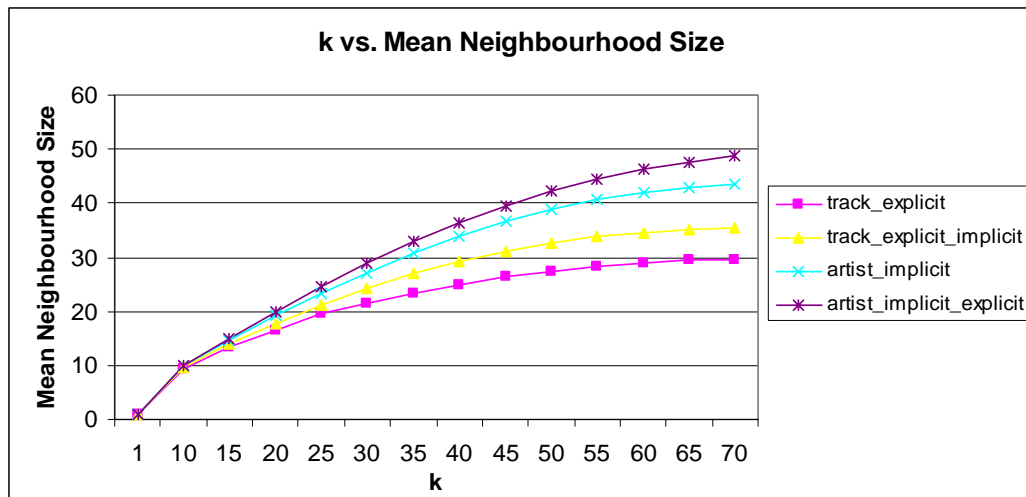


Figure 7.2: The graph plots mean neighbourhood size as the maximum neighbourhood size, k , is increased

Test 2: k vs. mean absolute error per user

Test 2 examines the effect of increasing maximum neighbourhood size, k , on the mean absolute error per user for each dataset. The value of k is incrementally increased and the mean MAE for each user is calculated. The graph in Figure 7.3 plots the overall mean MAE per user for each increment of k . For the purpose of comparison we include the baseline result from Table 7.5. Since

the baseline technique does not use neighbourhoods in its calculation the result remains constant for each increment of k .

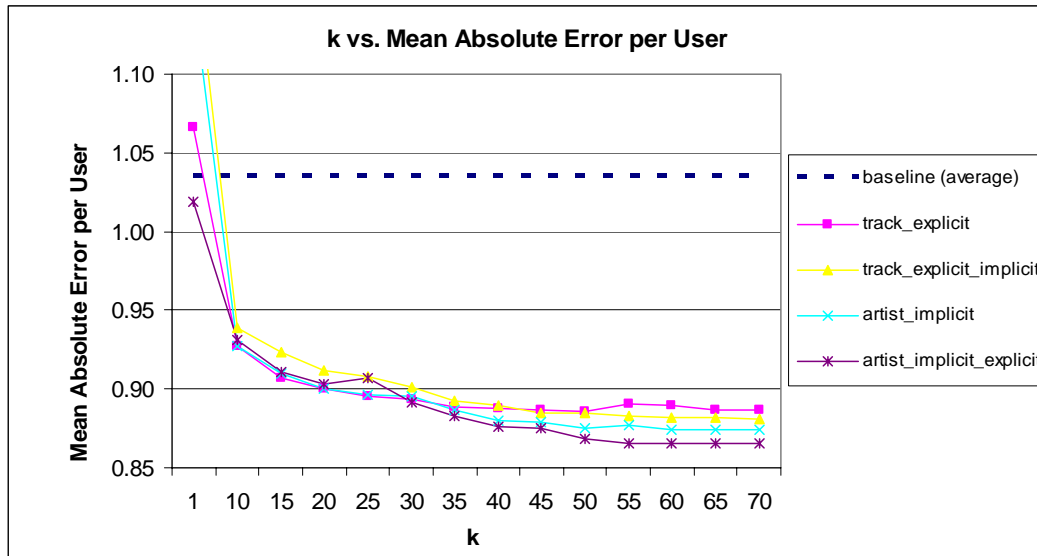


Figure 7.3: MAE vs. maximum neighbourhood size, k

The first thing we notice is that the MAE for all four datasets is relatively high compared to the MovieLens dataset. The higher figure on the Smart Radio data must be attributed to the *lack* of user instances in the Smart Radio dataset. While the Smart Radio dataset has 71 user instances with 4131 tracks to rate, the MovieLens dataset has 943 users with a possible 1682 movies to rate. Although datasets 1 and 2 are sparser than the MovieLens dataset, datasets 3 and 4 are considerably less sparse than the MovieLens dataset.

Essentially, the problem boils down to an inability to find a set of neighbours from a limited set of user instances that approximate each user's taste sufficiently. We can demonstrate this by examining the MAE for each user at a fixed value of k . Figure 7.4 plots the MAE at $k = 50$ for each user in the `track_explicit` dataset. The graph demonstrates that there are several users who are significantly above the mean MAE for $k = 50$. We find a positive correlation between *average neighbourhood size* and MAE for the `track_explicit` dataset (see Table 7.6), suggesting that the source of the error is poor neighbour *selection*. In effect, these users are making do with poor neighbours because there are no better available in the dataset. This is exacerbated by the sparsity of the dataset, as many users cannot be correlated at all because they have no items rated in common. When we plot the MAE for each user at $k = 50$ for the `artist_implicit_explicit`, we find that while the overall mean error rate is lower, many of the same outlying users still exist. However, there is very little correlation between MAE per user and average neighbour size (see Table 7.6), which suggests that the increased density of this

dataset allows each test user to find a better set of neighbours at $k = 50$ than is possible with the `track_explicit` dataset.

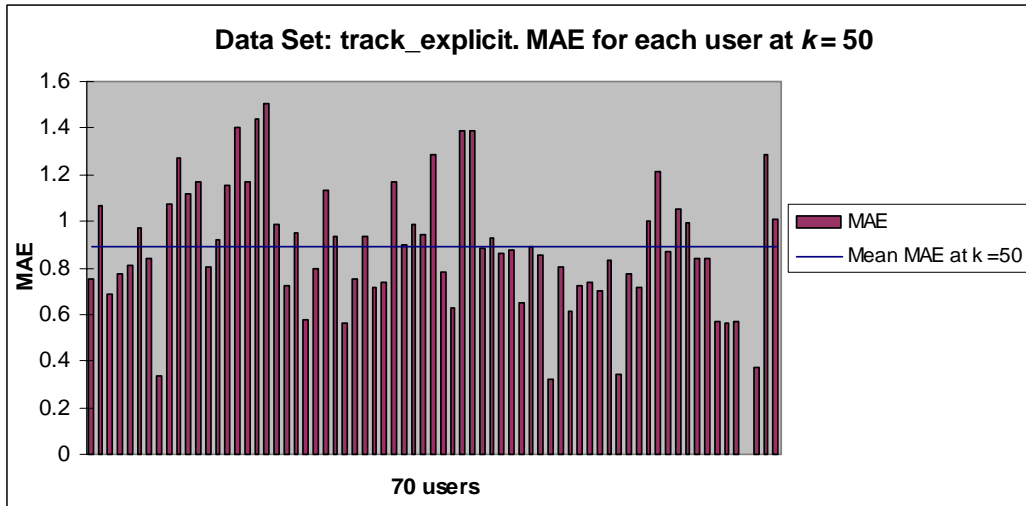


Figure 7.4: MAE for 70 users in the `track_explicit` dataset for $k = 50$. We could not make predictions for one user.

Table 7.6: Correlation between MAE per user and the average neighbourhood size

Dataset	Correlation: MAE vs. Mean Neighbourhood Size
<code>track_explicit</code>	0.132401
<code>artist_implicit_explicit</code>	0.050327

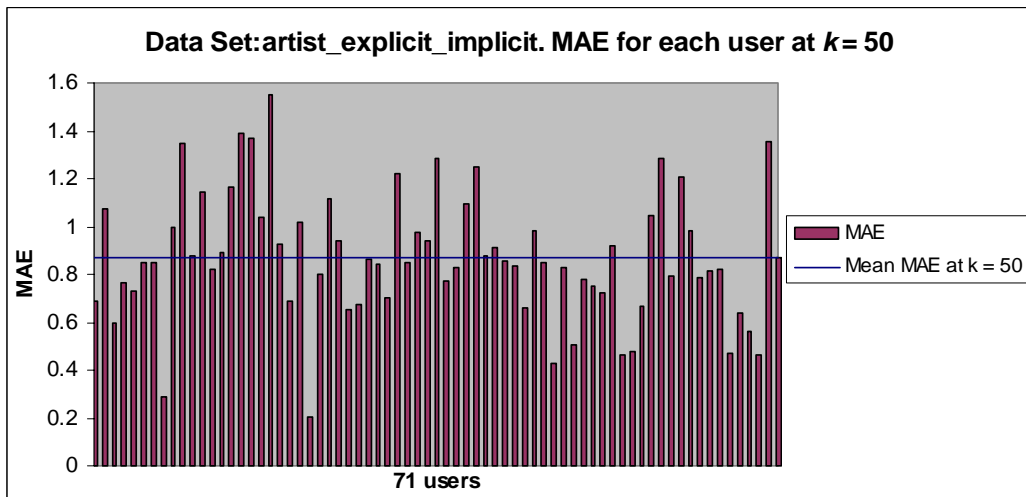


Figure 7.5: MAE for 70 users in the `artist_implicit_explicit` dataset for $k = 50$

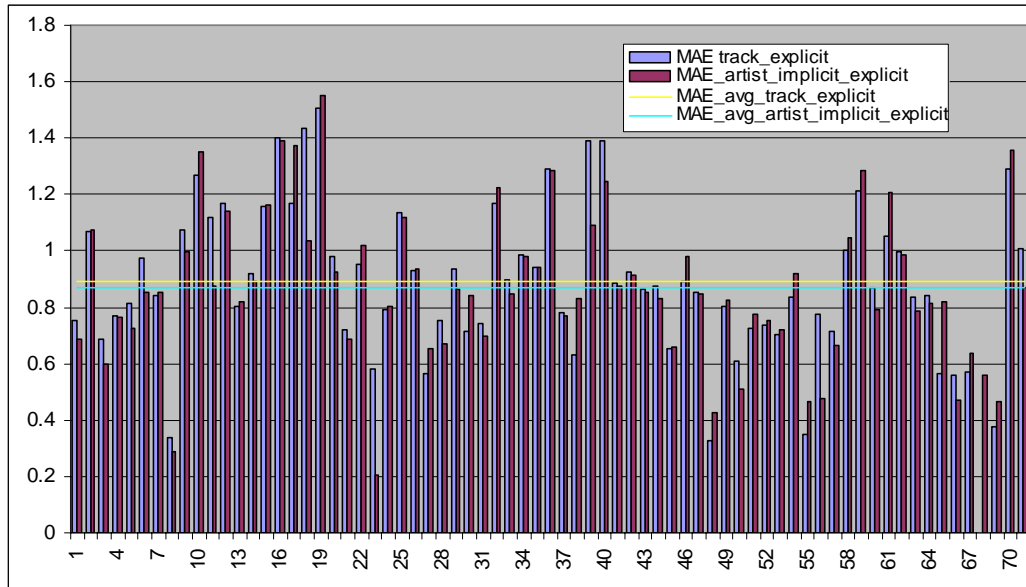


Figure 7.6: This graph overlays the graphs in Figure 7.4 and Figure 7.5

This hypothesis is strengthened by the shape of the graph in Figure 7.3, obtained by plotting MAE against k . The MAE drops as k is increased but does not increase again as k grows larger as has been demonstrated with the MovieLens dataset. This would suggest that the algorithm has insufficient *user* data to converge to an optimal mean value for k . However, achieving an optimal value of k is not necessary for the ACF algorithm to work. While experiments on the MovieLens dataset have suggested that a best- k neighbourhood selection performed optimally (Herlocker et al. 1999), Sarwar et al. (2000b) have observed that the technique for determining neighbourhood is dataset dependent. In the GroupLens project, which we discussed in Chapter 4, a weighted calculation over the entire dataset was used instead of a best- k technique (Resnick et al. 1994). This is essentially best- k with k set to the number of users available. Since prediction is made using a weighted average, the correlation weights perform the heavy lifting when it comes to prediction. Our experiments suggest that due to sparsity and lack of user data this is currently the best technique to use with the Smart Radio data. As the dataset grows we may need to set k to a lower level to extract optimal performance.

In any case, we can see from Figure 7.3 that using ACF on each dataset outperforms the baseline method. To test our results for statistical significance we need to examine the absolute error in the scores predicted with each dataset configuration at a particular value of k . To do this we use a paired t -test between the sets of scores predicted for each dataset configuration. Our first objective is to show that the result for ACF vs. the baseline is statistically significant. As Table 7.7 demonstrates, we find that this is the case for each dataset within a 99% confidence level.

Table 7.7: Smart Radio: baseline vs. ACF; paired t -test

Dataset Configuration	Results
baseline vs. ACF, $k = 70$ track_explicit	Alternative hypothesis: baseline \geq track_explicit Difference between means: 1.880739 99% CI : 1.523655 to $+\infty$ t statistic : 12.26 1-tailed p : <0.0001
baseline vs. ACF, $k = 70$ track_explicit_implicit	Alternative hypothesis: baseline \geq track_explicit_implicit Difference between means: 1.70951 99% CI: 1.343872 to $+\infty$ t statistic : 10.88 1-tailed p : <0.0001
baseline vs. ACF, $k = 70$ artist_implicit	Alternative hypothesis: baseline \geq artist_implicit Difference between means: 1.75336 99% CI : 1.402054 to $+\infty$ t statistic : 11.61 1-tailed p : <0.0001
baseline vs. ACF, $k = 70$ artist_implicit_explicit	Alternative hypothesis: baseline \geq artist_implicit_explicit Difference between means: 1.86512 99% CI : 1.507157 to $+\infty$ t statistic : 12.12 1-tailed p : <0.0001

While ACF on all datasets was shown to significantly outperform the baseline technique, the MAE scores (see Figure 7.3) between the datasets themselves are very close. We perform t -tests between each pair of datasets and find that many of the differences shown in Figure 7.3 are not significant.

For instance, the decrease in MAE per user for the `artist_implicit` dataset and the `artist_implicit_explicit` over `track_implicit` exhibited in Figure 7.3 is shown not to be statistically significant. This is hardly surprising since the `artist_` datasets are derived from the `track_explicit` dataset. The key observation, however, is that the `artist_` datasets do not significantly contribute to an increase in mean absolute error.

While MAE *per user* is shown to be slightly lower for `track_explicit_implicit`, in fact the opposite is the case when we take the MAE *per item*. A paired t -test suggests that the

`track_explicit_implicit` dataset configuration is marginally less accurate per test item than the `track_explicit` configuration with a confidence level of 95% (see Table 7.8). Again, this is not a surprising result given that the implicit scores in the dataset are derived using a heuristic measure.

The decrease in MAE for the `artist_implicit_explicit` dataset over the `artist_implicit` dataset is statistically significant with a confidence level of 95%, suggesting that the addition of the less noisy, explicit artist data improves the correlations between neighbours.

Table 7.8: Paired *t*-tests that demonstrate statistical significance between paired dataset configurations

Dataset Configuration	Results
ACF, $k = 70$ <code>track_explicit</code> vs. <code>track_explicit_implicit</code>	Alternative hypothesis: MAE <code>track_explicit_implicit</code> \geq MAE <code>track_explicit</code> Difference between means: -0.160649 95% CI : $-\infty$ to -0.039751 <i>t</i> statistic : -2.19 1-tailed <i>p</i> : 0.0144
ACF, $k = 70$ <code>artist_implicit</code> vs. <code>artist_implicit_explicit</code>	Alternative hypothesis: MAE <code>artist_implicit</code> \geq MAE <code>artist_implicit_explicit</code> Difference between means: 0.102232 95% CI: 0.004819 to $+\infty$ <i>t</i> statistic : 1.73 1-tailed <i>p</i> : 0.0422

Test 3: *k* vs. coverage

This test examines how coverage is affected as we increase *k*. We define coverage as the percentage of test items that the algorithm can make a prediction for. Our test set contains 7808 items sampled using the `Leave_One_Out` method. Our baseline achieved a prediction coverage of 94.6% which is the optimal coverage using the `track_explicit` data. Since there are 422 items (5.4%) for which there is one rating, no prediction can be made for these items using the remainder of the dataset. This is also the optimal prediction coverage for the `artist_implicit` and `artist_implicit_explicit` datasets, which are dimensionally reduced versions of the `track_explicit` dataset. However, we would expect the `track_explicit_implicit` dataset to have higher coverage because of the addition of the set of implicit ratings derived from our server logs.

As we can see from Figure 7.7, all four datasets converge with varying degrees of speed towards the baseline optimal coverage. The `track_explicit_implicit` exceeds the baseline score after k reaches 30.

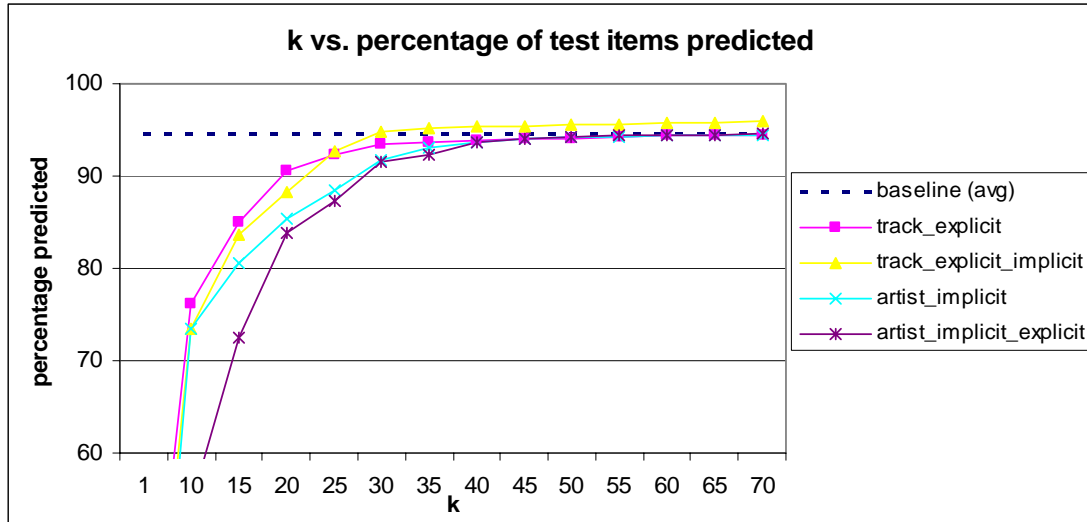


Figure 7.7: k vs. percentage of items predicted

The coverage issue cannot be separated from the measure of MAE at each increment of k . While prediction coverage may quickly reach a high level for all datasets, the underlying data informing each prediction is quite sparse, which contributes to the MAE at that value for k .

7.1.9 Prediction Correlation

In terms of prediction correlation, the baseline technique performed considerably worse on the Smart Radio data than the MovieLens data. This may be due to the lack of user data and the sparsity of the data where one or two ratings contribute to the mean score. Comparing the four datasets we see that the `track_explicit` and `artist_implicit_explicit` datasets perform optimally (see Table 7.9). The `track_explicit` dataset outperforms `track_explicit_implicit` with a confidence level of 95%, which we suggest is due to noise in the implicitly derived data. The `artist_implicit_explicit` dataset improves upon the correlation score achieved by the `artist_implicit` dataset, suggesting that the addition of the explicit artist data improves the accuracy of the correlations between neighbours. This result is observed with a confidence level of 95%.

Table 7.9: Prediction correlation scores for the baseline technique and ACF on the four datasets at $k=70$

Smart Radio: Prediction Correlation	
track_explicit,Baseline:	0.181979
track_explicit,ACF ($k = 70$)	0.408648
track_explicit_implicit,ACF ($k = 70$)	0.393767
artist_implicit, ACF ($k = 70$)	0.397517
artist_implicit_explicit, ACF ($k = 70$)	0.405527

7.1.10 Dimensionality Reduction

Our results find that dimensionality reduction by mapping music item scores to artist scores does not significantly alter the MAE. However using the artist dimension allows us to bootstrap new users using artist scores rather than ratings from individual items.

As we describe in Section 6.5.3, the key issue in bootstrapping a user into the system is to elicit the minimum amount of information necessary so that good correlations can be made with other users. By asking the users to rate seven artists, we have a much better chance of finding a set of neighbours than if we ask them to rate seven music tracks. In effect we achieve a better result by asking the user to perform the same amount of work. In fact our experiments suggest that adding explicit artist data to implicitly derived artist data decreases the MAE per user.

Another benefit of dimensionality reduction is the increased speed with which we can calculate correlations across 333 dimensions rather than 4131. This is particularly pertinent for large ACF datasets where real time predictions are required. Sarwar et al. have found that dimensionality reduction is a useful technique to reduce query-time for the calculation of neighbour users, but has poorer results for predictive accuracy (Sarwar et al. 2000a).

As we describe in Section 6.6.2, our implementation of ACF using a case retrieval net means that correlations can be calculated very quickly. In the figure below we plot the average prediction time per prediction at different values for k for the `track_explicit_implicit` dataset and the dimensionally reduced `artist_implicit_explicit` dataset.

On average, we can produce predictions twice as fast from the `artist_implicit_explicit` dataset than from the `track_explicit_implicit` dataset. One advantage of an increase in prediction speed is that bootstrap recommendations, which are calculated and presented synchronously, can be delivered very quickly.

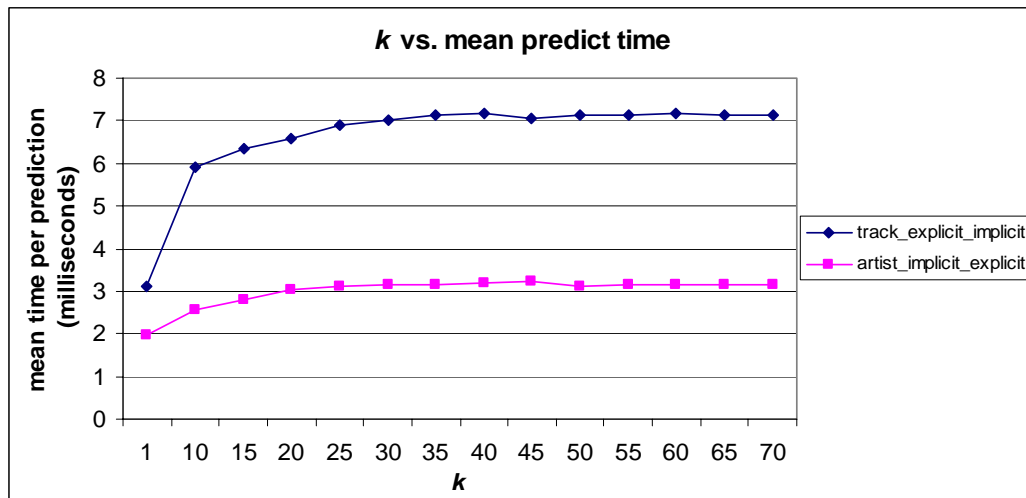


Figure 7.8: k vs. the mean time per prediction

7.1.11 Conclusions from Off-line Analysis

We have shown that ACF performed on the Smart Radio datasets exhibits a greater MAE per user than the MovieLens dataset. However, the Smart Radio dataset is considerably sparser than the MovieLens dataset, and our experiments would suggest that there is not enough data to demonstrate an optimal value for k . However, the determination of optimal value for k has been shown to be dataset dependent (Sarwar et al. 2000b), and in the case of Smart Radio we need to apply the technique of the GroupLens project where all correlated users were used to produce a prediction.

We have found that the `track_explicit` and `artist_implicit_explicit` datasets perform best. The advantage of using the `artist_implicit_explicit` data is that it allows us to quickly bootstrap new users, and has lower memory and computation costs when calculating predictions. However, our findings must be tempered by the fact that we are working with less data than we would like. These results may change significantly as the dataset grows. However, tracking the parameters that influence algorithmic performance on each particular dataset is the key role of off-line analysis in a system like Smart Radio.

Which dataset to use?

Initially, Smart Radio used only explicit track data as the basis for correlating users (`track_explicit` dataset). Data sparsity meant that we had problems correlating users to make recommendations. We realised that adding implicitly derived data allowed the system to correlate more users, and for a time we used the `track_explicit_implicit` dataset, even though it gave us marginally lower prediction accuracy. While experimenting with improving the bootstrap facility, we developed the `artist_implicit_explicit` dataset which is now

used as the basis for calculating correlations in Smart Radio. The system, however, can fall back on the `track_explicit_implicit` dataset where necessary. This occurs where the user has not explicitly rated a track or an artist. A portion of users who joined Smart Radio before the new bootstrap facility would fall into this category. For these users, we use the `track_explicit_implicit` dataset as the basis for calculating correlation with other users. However, where a user has not explicitly rated a track or artist his/her data is not considered when calculating a neighbourhood set for another user. As we describe in Section 6.4.1, this is because, in such cases, the implicit scores are noisy, being based on the average explicit scores in the dataset.

Smart Radio is accumulating an increasing amount of implicit track data. Although, our implicit data algorithm is satisfactory, we feel that a more precise way of deriving implicit scores is possible. For instance, if we could capture data directly from the player device such as track skips, volume changes, duration of listening – we could possibly produce better implicit scores.

7.2 On-line Evaluation

Our on-line evaluation section consists of two parts. The first is concerned with examining whether the recommendation strategy as a whole was successful. We do this by examining the source of each playlist that was played during the evaluation. By ‘source’ we mean the section of Smart Radio from which the playlist was chosen. There are six possible sources: ‘top lists’, ‘past playlists’, ‘trusted neighbour’, ‘recommendations’, ‘explicit search’ and ‘compiled from scratch’. In this part of the on-line evaluation we would hope to show that the recommendation section was a source that users regularly turned to when finding playlists. In the second part of the on-line evaluation we look at the recommendation strategy in isolation. During the evaluation period two recommendation strategies were deployed concurrently: ACF and context-boosted ACF. The goal of our analysis is to demonstrate which strategy was preferred by the Smart Radio users.

7.3 Recommendations vs. Other Sources of Playlists

The first part of our analysis looks at what features of the Smart Radio system users tended to use when selecting playlists to play. Our results refer to the listening data of 58 users who played a total of 1012 playlists during the 101 day period from 08/04/2003 until midnight 17/07/2003.

Table 7.10: A summary of the on-line dataset we collected during our evaluation period

On-line Evaluation Dataset Summary	
Description: dataset consisting of four fields: <i>userId</i> , <i>playlistId</i> , <i>source</i> , <i>date</i>	
Number of instances	1012
Number of users	58
User average	17.45
User standard deviation	30.97
Possible values for the source field	"toplists", "pastlists", "trusted_neighbour", "recommendations", "explicit_search", "compile_from_scratch"

The graph in Figure 7.9 shows the source of playlists played in the system for this period. The cumulative scores for that period would suggest that 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 somewhat unusual as users are initially required to choose a playlist from one of the other categories before they can choose it again as a past playlist. In this case, 23 of the past playlists chosen were originally recommendations made within the period. We should also note that building playlists from scratch or explicitly searching for playlists cannot be considered ‘rival’ categories to the recommendation category. As we described in Chapter 4, an ACF recommender requires users to discover a proportion of new items from outside the recommendation system itself. Were users to stop building new playlists, the ACF recommender would dry up, having no new items to recommend. Therefore, while an ACF recommender strategy should alleviate the burden of manually searching or filtering, it cannot remove it completely. Users have to input some work in order for the ACF system to distribute the benefits to others. No research has been done on describing the parameters affecting the minimum ‘work sink’ for an ACF system. In this evaluation, we wish to demonstrate that the recommendation strategy appears to be succeeding, with just under 50% of playlists originating there.

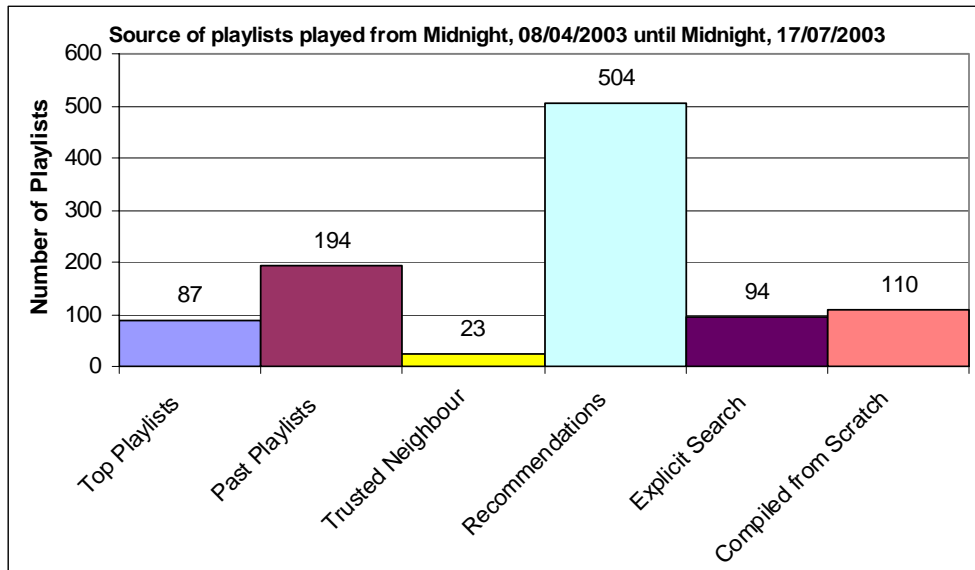


Figure 7.9: The source of playlists played in Smart Radio during the evaluation period

7.3.1 Weekly analysis

In order to check that the performance is consistent over the evaluation period, we divide the 101 days into 15 intervals: 14 seven-day intervals plus a remainder of 3 days. We considered an interval length of one week was the most appropriate since it would reflect the working patterns of our users. For instance, usage of the system died off dramatically at weekends when most of our listeners were not at work.

Table 7.11 tabulates the results for these weekly intervals. We can see from Figure 7.10 that the proportion of recommendations in relation to the other means of finding playlists remains high during the weekly intervals.

Table 7.11: Source of playlists: weekly breakdown: 08/04/2003 until midnight 17/07/03

Week	Top Playlists	Past Playlists	Trusted Neighbour	Recommendations	Explicit Search	Compiled from Scratch
1	9	6	1	24	3	3
2	2	9	1	15	1	3
3	5	6	1	24	1	3
4	6	6	0	18	3	7
5	6	9	0	27	5	13
6	8	17	2	39	5	8
7	7	10	0	42	6	6
8	5	17	2	46	6	11
9	3	16	5	48	8	15
10	2	5	9	33	4	8
11	3	12	1	46	15	7
12	9	9	0	33	8	5
13	16	29	1	51	16	8
14	5	23	0	34	7	9
15	1	20	0	24	6	4

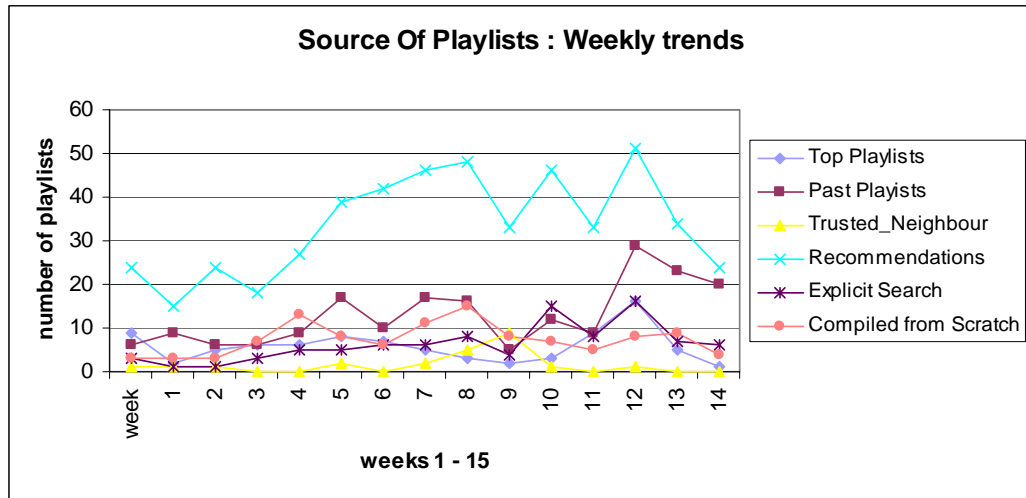


Figure 7.10: Source of playlists: weekly trends

7.3.2 User Trends

As well as a time-based analysis we carry out a *user*-based analysis to determine whether the behaviour suggested by Figure 7.9 and Figure 7.10 is true for a significant proportion of our users.

A cursory inspection of our dataset reveals that during the evaluation period we had users who used the system several times a week 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 how many playlists each user has listened to. We have 10 categories in which users may fall, representing different degrees of usage of the system. Each category represents a range of playlist usage. The range categories and the number of users per category are presented in Table 7.12. The left column of the table presents a series of ranges representing the number of playlists a user has chosen to play during the evaluation period. These are qualified as *light*, *medium* and *heavy* usage categories. The two right most columns give the number of users that fall within these ranges, and the number of playlists played by these users respectively. The information within this table is represented visually in Figure 7.11 and Figure 7.12.

We can see from Figure 7.11, that 39 users (67%) were *light* users that played from 1 to 10 playlists, 14 users were *medium* users (24%) who played between 10 and 50 playlists, and 5 users were classed as *heavy* users (9%) who played in excess of 50 playlists. If we view Figure 7.12 we can see that the majority of the playlists are assigned to users that lie in the *medium* and *heavy* usage range (88%). We need therefore to examine whether a few prolific users are not overly boosting the trends we see in Figure 7.9 and Figure 7.10.

Table 7.12: The different degrees of usage of the system during the evaluation period

Range: total playlist played		Number of users	Number of playlists
1-5	Light Use	32	67
6-10		7	53
11-15	Medium Use	3	37
16-20		2	38
21-30		5	128
31-40		1	32
41-50		3	132
51-60	Heavy Use	0	0
61-80		3	237
81 +		2	288
Total		58	1012

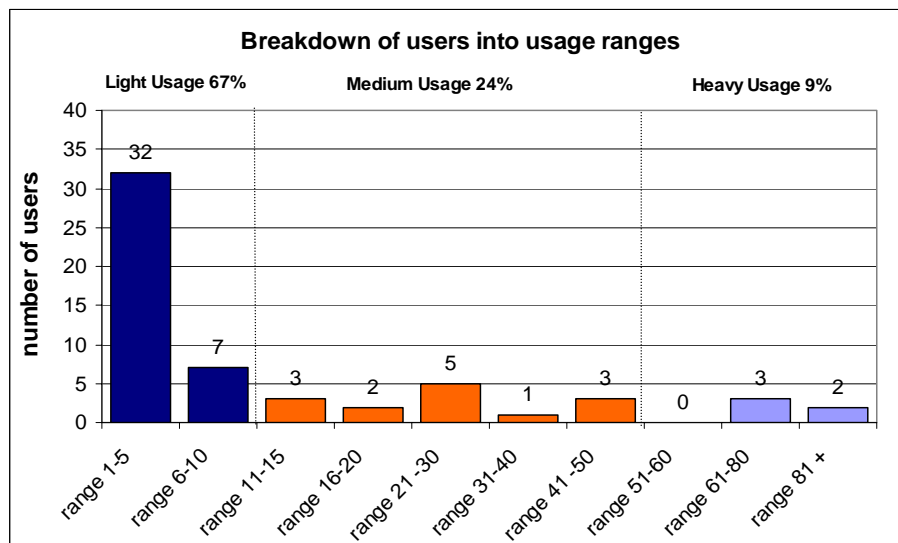


Figure 7.11: The graph illustrates that the majority of users were *light* users, playing from 1 to 10 playlists

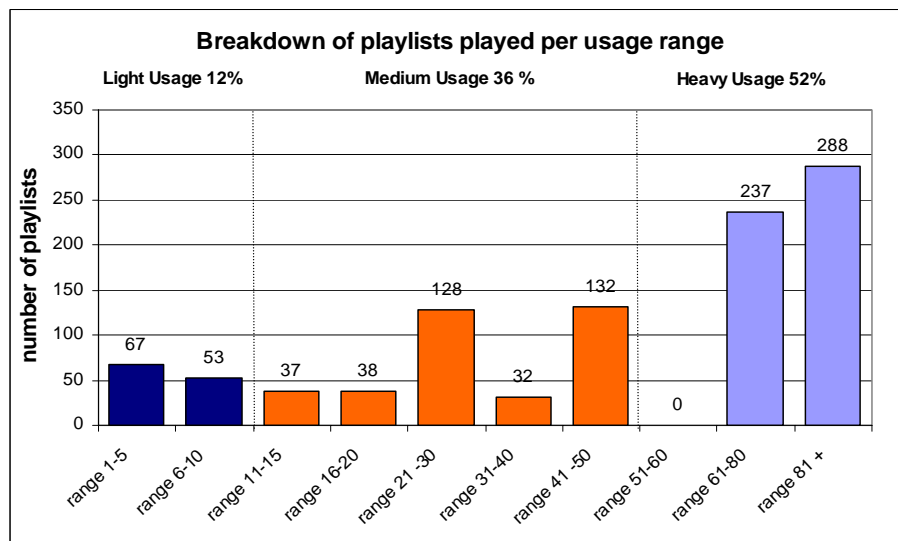


Figure 7.12: The breakdown per usage range of the 1012 playlists played during the evaluation period.

The graph in Figure 7.13 illustrates the proportions of playlist sources in each usage range. The percentage figure indicates the percentage of the total in each range that came from the *recommendations* category. The recommendation strategy is, in fact, proportionally in keeping with the cumulative mean, 49%, across all the usage ranges.

However, one cause for concern is the fact that out of 58 users, 44 took advantage of the recommendation strategy, whilst 14 chose to ignore it completely (see Table 7.13). This fact has been hidden in the data analysis up to this point. On closer inspection we can see why: 13 of these 14 users are very *light* users indeed (a mean of 1.54 playlists per user). The 14th user is an outlier in the sense that this user repeatedly played the same three playlists, and does not choose from any other category other than ‘past playlists’ (see Table 7.14). Thus we need not worry about these user instances. For most of the 13 *light* users they simply had not used the system long enough to engage with the recommendation facility.

Table 7.13: The number of users who did and who did not make use of the recommendation strategy

Total number of users	58
Number of users who used recommendations	44
Number of users who did not use recommendations	14

Table 7.14: The usage range for users who did not use any recommendations for the evaluation period

Usage range for users who did not use recommendations	
Range	Number of Users
1-5	13
41-50	1

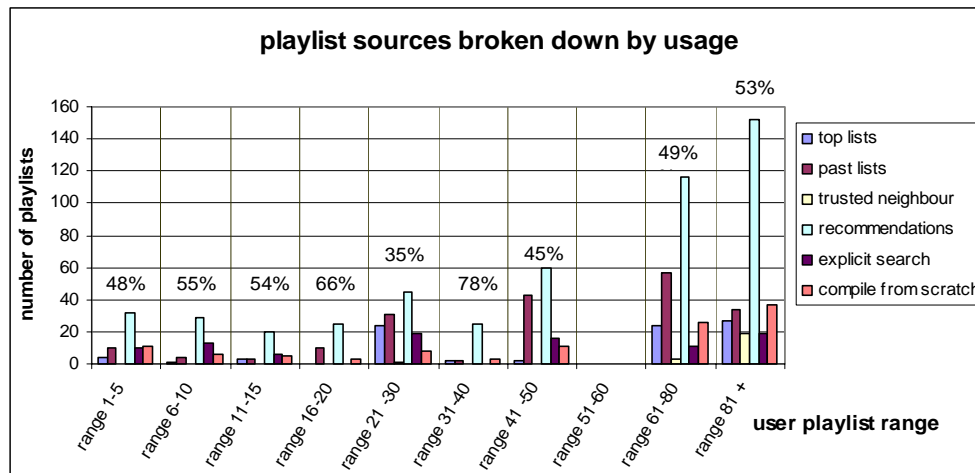


Figure 7.13: The proportions of playlist sources in each usage range

7.4 ACF vs. Context-Boosted ACF

The previous section indicated that a high percentage of playlists originate from the recommendation strategies employed by Smart Radio. In this section we introduce a methodology to compare the efficacy of two recommendation strategies that cannot be compared using an off-

line analysis. We wish to examine whether context-boosted ACF is preferable to plain ACF in the experience of our users. The key question is whether our recommendations actually help the user exploit the resources available within the larger application domain. When employing a recommender system, the goal is to translate continued user satisfaction into continued use of the system resources. It is the continuing use of these resources based on the advice of the recommender that requires analysis when conducting an on-line evaluation of a recommender system.

How does a recommendation strategy maintain user satisfaction? Generally, a recommendation engine will present a list of n resources based on the user's feedback or a user profile. The simplest way to do this is to present an ordered list of recommendations based on a score calculated by an algorithm. This score might reflect the similarity of a resource to a user profile/query or a score predicted by a collaborative filtering algorithm. However, a recommendation engine may choose other strategies to present a relevant list of resources to the user. Smyth and McClave (2001) show that a recommendation set that contains too many similar items may be highly redundant whereas a small, diverse set would offer the user more choice. Swearingen and Sinha's research (2001) would suggest that including "trust generating items" in a recommendation set is perceived as being "useful" by the user. In Chapter 5, we introduced a notion of context-boosting where recommendations are ranked according to their relevance to the user's current listening preferences. The common factor for these approaches is that they provide *enhancements* to an existing approach that cannot readily be evaluated in an off-line manner.

7.4.1 On-line Evaluation Methodology

In Section 7.1 we described an evaluation of a recommender system from a machine learning perspective that treated the recommendation task as a regression. With such an approach we can examine the limits within which the ACF algorithm will work on a particular dataset. However we cannot use this technique to evaluate an enhancement that produces a ranking based on user actions at a particular time. To be sure that our new hypothesis is working it is important we perform a comparative analysis of how it performs *against* a pure ACF strategy.

We do this using an on-line evaluation in which 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 another in a live application and measures the relative degree of success of each strategy. Our evaluation methodology draws upon an on-line evaluative framework for recommender systems that we have earlier defined (Hayes et al. 2002b). This methodology does not allow us to measure absolute user satisfaction, but only *relative* user satisfaction with one recommendation strategy over another.

7.4.2 Evaluation Environment

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 context-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 contest between two different approaches to solving the same problem (in this case, winning user satisfaction) in which the outcome is defined by whether the user makes use of recommendations. We define three evaluation policies:

Presentation policy:

The recommendations in Smart Radio are presented as an ordered list. For evaluative purposes we *interleave* items from each strategy into a **merged set** (see Figure 7.14). 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 a playlist is selected to play, which is the event that triggers the context ordering of the ACF recommendations.

Evaluation policy:

We need to define 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 first selecting it from the recommendation set. A user may choose to adapt and then play a recommended playlist. We still consider this evidence of a preference for the chosen playlist over the others available.

Comparison metric:

The 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 of each system over time rather than simply using a cumulative score. We also need to look at the distribution of successes over the user population.



Figure 7.14: A screenshot illustrating an interleaved recommendation set

7.4.3 Results

Figure 7.15 illustrates the cumulative breakdown of recommendations between pure ACF recommendations and context-boosted ACF for the period. From a total of 504 recommendations that were played, 311 were sourced from context-boosted recommendations, 177 came from normal ACF recommendations and 16 were bootstrap recommendations.

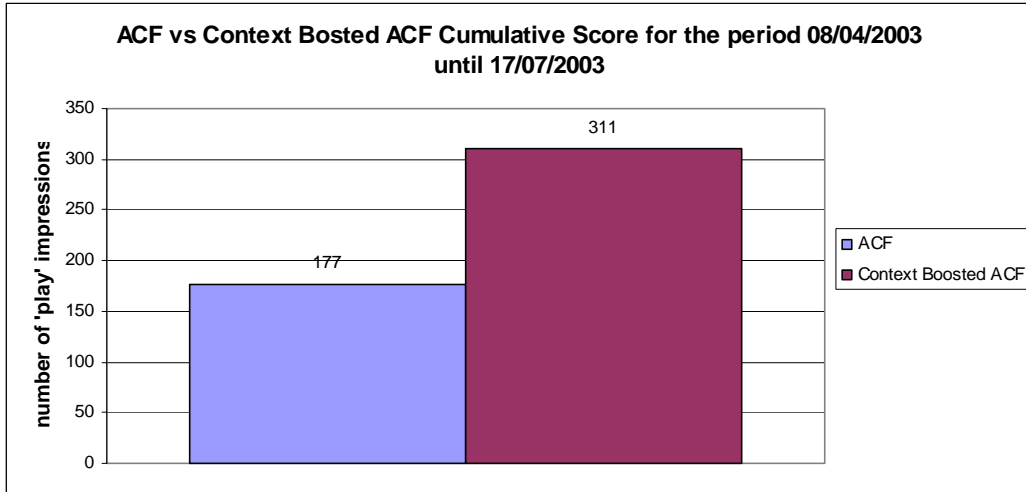


Figure 7.15: The cumulative scores for the ACF vs. context-boosted ACF analysis

In order to check that these results were consistent throughout the evaluation period we analysed our data over 15 weekly intervals. The graph in Figure 7.16 shows the proportion of ACF to context-boosted recommendations, analysed on a weekly basis for the period. In all but one interval the context-boosted ACF outperformed the pure ACF recommendation strategy. Table 7.15 gives the results of a paired *t*-test that confirms this result to be statistically significant with a confidence level of 99%.

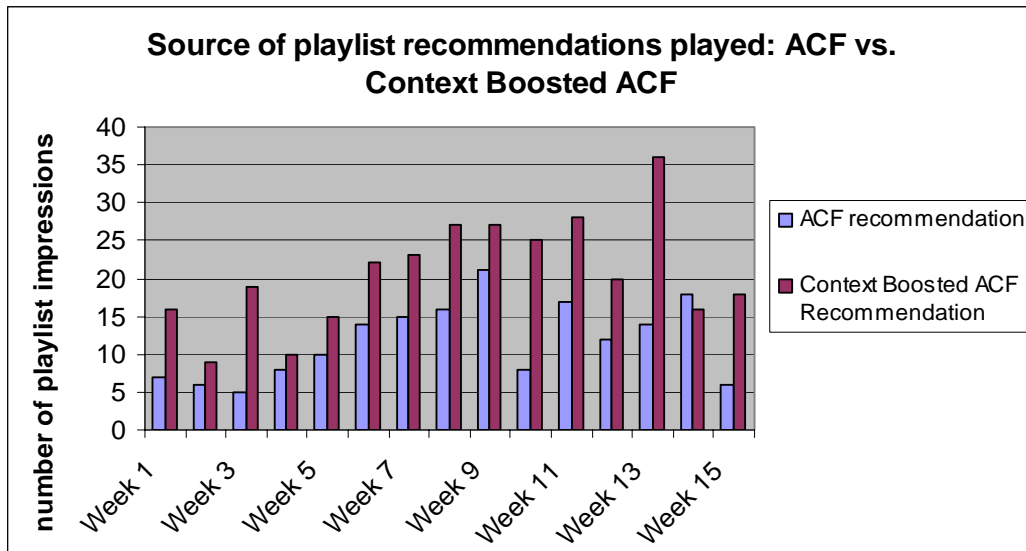


Figure 7.16: ACF vs. context-boosted ACF over a 15-week period

Table 7.15: ACF vs. Context-boosted ACF analysed over 15 weekly intervals. 1-tailed *t* test with CI of 99%.

Test: Paired <i>t</i> -test				
ACF. Vs Context-boosted ACF per Week				
Alternative hypothesis ACF Types: ACF ≤ Context-boosted ACF				
<i>n</i> 15				
ACF Types	<i>n</i>	Mean	St. Dev	St. Error
ACF	15	11.8	5.0	1.30
Context-boosted ACF	15	20.7	7.2	1.87
Difference	15	-8.9	6.0	1.56
Difference between means	-8.9			
99% Conf. Interval	-∞ to -4.8			
<i>t</i> statistic	-5.74			
1-tailed <i>p</i>	<0.0001			

User analysis

As we saw in Figure 7.11 and Figure 7.12, the majority of our playlist data originates from *medium* and *heavy* users of the system. Since these users are not evenly distributed in the data we need to examine the proportions of ACF to context-boosted ACF over the usage ranges established earlier. Figure 7.17 illustrates the results. Whilst ACF is marginally greater in two intervals, if we use a paired *t*-test to test these results for significance, we find that the hypothesis, $ACF \leq Context\text{-}boosted\ ACF$ holds with a confidence level of 95% (see Table 7.16).

If we perform another t -test, this time on the individual user recommendation data (represented by the graph in Figure 7.18), we find that the hypothesis, $ACF \leq \text{context-boosted ACF}$ once again holds with a confidence level of 95% (see Table 7.17).

Inspection of Figure 7.17 and Figure 7.18 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.

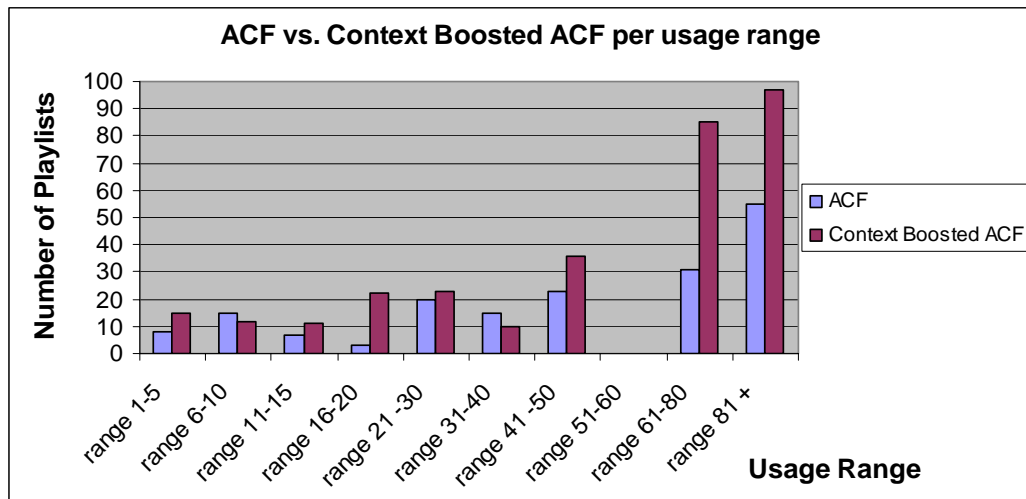


Figure 7.17: ACF vs. context-boosted ACF divided into ranges of usage

Table 7.16: A paired t -test on ACF vs. context-boosted ACF for the usage ranges shown in Figure 7.17

Test:		Paired t -test			
Alternative hypothesis		ACF vs. Context-boosted ACF: per usage range			
n		ACF Type: $ACF \leq$ Context-boosted ACF			
ACF Type		n	Mean	St. Dev.	St. Error
ACF		10	17.7	16.2	5.11
Context-boosted ACF		10	31.1	33.1	10.47
Difference		10	-13.4	19.8	6.26
Difference between means		-13.4			
95% Conf. Interval		-∞ to -1.9			
t statistic		-2.14			
1-tailed p		0.0304			

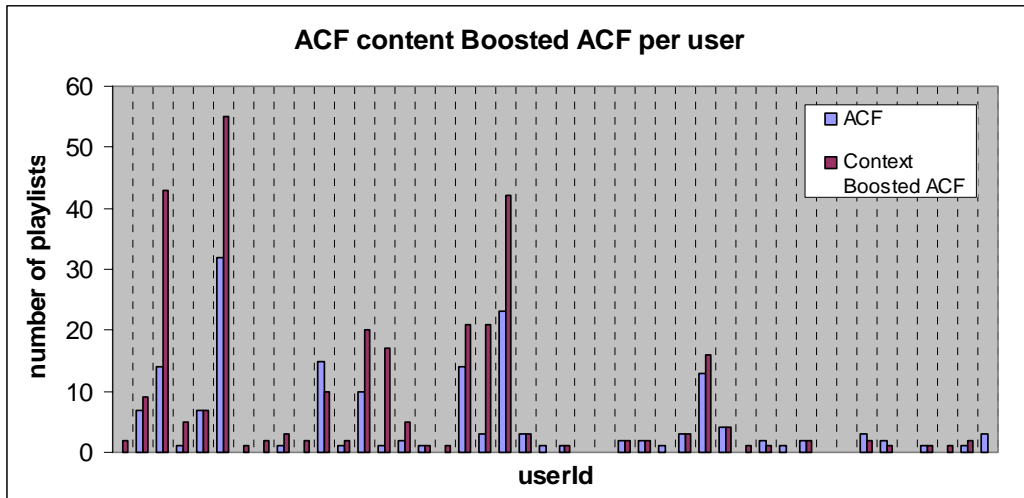


Figure 7.18: The proportion of ACF to context-boosted ACF per user during the evaluation period

Table 7.17: ACF vs. context-boosted ACF. Results of a paired *t*-test on individual user recommendations.

Test	Paired <i>t</i> -test			
	ACF vs. Context-boosted ACF: per User			
Alternative hypothesis	ACF Type: ACF \leq Context-boosted ACF			
<i>n</i>	39			
ACF Type	<i>n</i>	Mean	St. Dev.	St. Error
ACF	39	4.5	6.9	1.11
Context-boosted ACF	39	8.0	12.9	2.07
Difference	39	-3.4	7.4	1.19
Difference between means	-3.4			
95% Conf. Interval	$-\infty$ to -0.5			
<i>t</i> statistic	-2.89			
1-tailed <i>p</i>	0.0032			

7.4.4 Conclusions from On-line Analysis

The goal of the on-line evaluation was to determine a measure of relative user satisfaction with one recommendation strategy above another. We concurrently deployed two recommendation strategies, taking care to give each technique access to the first recommendation slot. Our experiment was conducted over a 101-day period in which 58 people used the system. We firstly examined whether the positive cumulative result in favour of context-boosted ACF would also hold over 15 weekly intervals. We found that it did.

Examination of our data showed that it was not evenly distributed amongst users, but that the majority of playlists were played by *medium* and *heavy* listeners. Since the majority of our listeners were *light* listeners, our analysis had to examine whether our results were distributed amongst these users as well as the more active users. Our results would indicate that while heavy users are more likely to choose the context-boosted ACF strategy, lighter users also demonstrated a preference for this technique.

7.5 Conclusion

In this chapter we demonstrated two techniques for evaluating our recommender system: an off-line and an on-line approach. An off-line approach is useful for determining whether our algorithms are performing according to previously observed norms, and for providing an insight into which strategy or dataset may perform optimally when used in a live system.

Our off-line analysis would indicate that the Smart Radio dataset does not have enough user data. For that reason we were unable to determine an optimal value for k . However, this is not necessary for the algorithm to work, and we demonstrated that four Smart Radio datasets could easily outperform a baseline averaging technique. We found that the `track_explicit` and `artist_implicit_explicit` datasets performed optimally. The latter dataset is created by transforming the user–music item matrix to a user–artist matrix using a simple transformation. Reducing the dimensions of the dataset from 4131 features to 333 artist features allows us to quickly bootstrap new users. Additionally, it has benefits for lowering memory costs and computational expense. However, off-line techniques are not suitable for measuring the efficacy of a recommendation strategy such as context-boosted ACF in which a ranking is produced based on user actions at a particular time.

In the second part of this chapter we described an on-line experiment in which we performed a comparative analysis of how context-boosted ACF performs against a pure ACF strategy. Unlike the off-line analysis, the on-line methodology judges the relative degree of success of each strategy according to whether the recommendations of each strategy were employed by the on-line user. The key idea is to have two recommendation algorithms simultaneously compete to provide the user with recommendations. We interleave recommendations so that each technique has an equal opportunity for success. We log the user’s response to the recommendations provided by each strategy and analyse the results. In our analysis we examined the results over 15 weekly intervals and found context-boosted ACF to be significantly more popular than ACF with a confidence level of 99%. We also analysed the data according to the usage patterns of our users. Many users were light users of the system and thus did not contribute very many playlists to the dataset. We found, however, that the preference for context-boosted ACF was general amongst heavy and light users of the system with a confidence level of 95%.

While this technique can only gauge relative user satisfaction with one recommendation strategy over another, it does allow the developer to test whether a new strategy is perceived as being useful to the user population.

Chapter 8: Conclusions and Future Work

8.1 Overview

The success of file-sharing networks demonstrates that there is a huge potential market for digital music services, if the music industry can find a service model that is attractive to listeners, and at the same time reward those involved in producing recorded music. The new music services being promoted by the music industry seek to duplicate the centralised physical distribution model and profit margin associated with this. While the industry claims that such margins are necessary to cover the cost of 85% of their roster of artists that are not profitable, such accountancy does not stand up for digital distribution, where there are no production and packaging costs, and where music can be ordered and delivered on demand. One source of expense for the music industry is marketing and promotion. The concept of digital distribution has always been attractive to the music industry, because it suggests the possibility of a direct marketing and distribution access to their customer, thus dis-intermediating several links in the physical value chain.

In this thesis we have presented a prototype for an alternative distribution approach, one in which the consumers themselves are producers, building playlists of music which are then passed on to like-minded listeners. In this approach the service provider is an enabler of community and social processes, allowing information and recommendations to flow between users of the system, rather than engaging in targeted marketing.

8.2 Content Issues

We provide a detailed description of the issues relating to retrieval and recommendation of digital audio files. Whereas information retrieval systems use keyword extraction to represent the subject matter of a document for indexing and retrieval, the comparable process for audio artefacts is much more difficult. Low-level signal analysis can represent the audio piece in terms of its component frequencies, but the process requires human intervention and modelling to develop higher level representations that can be used in retrieval systems. The alternative is to rely on human input for content-description which can be a knowledge-intensive process. We call this difficulty in procuring description for music *the content elicitation bottleneck*.

In Case-Based Reasoning (CBR) terminology this represents a lack of readily available *vocabulary* knowledge from which case representation and similarity metrics for retrieval can be developed. However, we note that there are varying strengths of CBR, which generally depends on the domain knowledge available. In the Smart Radio domain, we are able to build a light CBR system using the inexpensive but basic content descriptions extracted from the id3 tags in each

mp3. We use this, not as a primary means of retrieval, but to augment our Automated Collaborative Filtering algorithm.

8.3 Extending ACF

Automated Collaborative Filtering (ACF) offers an alternative to a content-based strategy like CBR. It can be viewed as a distributed content elicitation process where the content is numeric rather than symbolic, indicating the users' subjective judgements on a particular item. One of the advantages of ACF is that it implicitly captures the subtle distinctions people make between apparently similar items using criteria, such as aesthetic merit, which are hard to encode formally in a content-based system.

In Smart Radio we use the ACF facility as a recommendation strategy but also as the primary means of organising our listening community. Smart Radio listeners receive recommended playlists from their nearest neighbours every time they log in using the ACF strategy. These playlists will change based on the feedback they give the system. A major drawback to ACF is that it is unable to make recommendations that are suitable for the user's context. In a real time recommender system like Smart Radio, it is important that the system can quickly respond to the user's listening interests. We proposed solving this by using a case-based technique, which would allow us to rank ACF recommendations according to their similarity to the user context. Rather than describing individual songs in terms of their most salient features, our case representation describes the *composition* of a playlist in terms of the crude meta-tags we have been able to extract from the mp3 files. The semantic captured by this playlist representation is the mixture of types of music in the playlist. Although features such as *genre* are not particularly discriminating, our view is that these feature mixtures may capture at the *surface* level the intention of the playlist compiler. *Similarity*, in this scenario, refers to how alike playlists are in terms of their composition. Although our CBR strategy is of the weak variety, as described in Chapter 3, we feel this is completely appropriate for an extension to ACF, which is often used in domains where content description is scarce.

We have also explained three other techniques for working with music content. These techniques are influenced by our view that a recommendation system must cater for how people use the assets being recommended. The *novelty* weight allows users to bias the recommendation engine towards their preference for new music. The *playlist re-use window* allows previously heard playlists to be recommended again, if the system finds that the predicted score of the top-rated playlist to be too low. A *refractory* period prevents playlists containing recently played tracks from being offered to the user. ACF latency is addressed using three techniques: *trusted neighbour filtering*, *querying by instance* and *query-based search*.

8.4 On-line Evaluation

We demonstrated two techniques for evaluating recommender systems: an off-line and an on-line approach. Off-line approaches are generally used to determine whether an algorithm is performing according to previously observed norms, and for providing an insight into which technique may perform optimally when used in a live system. Our analysis showed that we were unable to determine an optimal value for neighbourhood size, k , which would indicate that the Smart Radio dataset is lacking user data. However, this is not necessary for the algorithm to work, and we showed that ACF on four Smart Radio datasets could easily outperform a baseline averaging technique. One of these datasets was created by transforming the user–music matrix to a user–artist matrix without loss of predictive accuracy. We show that reducing the dimensions of the dataset from 4131 features to 333 artist features is beneficial during the bootstrap phase, allowing the recommendation engine to produce better predictions when the user has submitted few ratings. In addition, dimensionality reduction lowers memory costs and computational expense during neighbourhood calculation.

However, off-line techniques are not suitable for directly comparing recommendation strategies that produce different types of ranking, such as context-boosted ACF in which a ranking is produced based on an analysis of the user’s current interests. We described an on-line experiment in which we performed a comparative analysis of how context-boosted ACF performs against a pure ACF strategy. Unlike the off-line analysis, the on-line methodology judges the relative degree of success of each strategy according to whether playlist recommendations were used by the on-line user. In this experiment we have two algorithms simultaneously compete to provide the user with recommendations. So that each technique has an equal opportunity for success, we present recommendations to the user as an interleaved set in which each algorithm has access to the top recommendation slot in a round-robin fashion. Our experiment, conducted over 15 weeks, found context-boosted ACF to be significantly more popular than ACF within a confidence interval of 99%. We also analysed the data according to the differing usage patterns of our users. We found that the preference for context-boosted ACF was general amongst heavy and light users of the system within a confidence interval of 95%. We suggest that, while this technique can only gauge relative user satisfaction with one recommendation strategy over another, it does allow the developer to test whether a new technique is perceived to be useful to the user population.

8.5 A Case-Based View of ACF

We demonstrate that the ACF mechanism is similar to an interactive CBR technique called *case completion* in which an incomplete target query is incrementally filled out in an interactive dialogue with the user. We view the ACF technique as a long-lived case completion technique in which the system uses the set of neighbouring user profiles to suggest case completion steps

(recommendations) to the user. Unlike typical CBR, however, the dialogue between the system and the user is on-going and case (user profile) completion is indefinitely postponed.

Using this insight we have addressed some of the problems inherent in ACF systems such as query efficiency and memory overhead by applying techniques used in CBR. We have implemented our ACF component using a Case Retrieval Net which reduces memory overhead by storing each feature-value pair once as an information entity (IE). We demonstrated how the calculation of the Pearson coefficient can be distributed among the IEs of the CRN during retrieval time. It has been recognised in the ACF literature that the use of this measure is a computational bottleneck when calculating neighbours from a large set of users. Accordingly, we showed how retrieval time is speeded up because local similarity calculations are performed once for each IE and distributed to indexed cases. Since the structure of CRNs can be easily adapted for parallel processing, the technique we propose can be scaled for much larger ACF data sets.

8.6 Perspective and Future Work

The Smart Radio system continues to be used in the Computer Science Department. We view the current system as a working prototype. In the following sections, we address issues relating to scalability and future development of the system.

8.6.1 Music Database

Smart Radio currently operates with a limited music database of 4,100 tracks. The differing genres of music in Smart Radio are not evenly distributed, and tend to reflect the interests of the colleagues and friends who contributed music to the project. Therefore, Smart Radio is biased towards producing more diverse recommendations from certain genres than others. For instance, *country* and *reggae* artists are not widely represented in the database. Therefore, there is a limited number of playlists in the system that could satisfy the tastes of fans of this type of music, and we would expect such users to receive unsuitable recommendations once they have exhausted the supply of those playlists. However, were a system like Smart Radio deployed with a database containing hundreds of thousands of tracks, with a greater selection of music from each genre, we would expect recommendations suited to each genre type to be generated. Of course, this relies upon correlating users from a potentially very sparse dataset. In such a scenario, dimensionality reduction techniques are imperative. In Smart Radio, we demonstrated how reducing the horizontal dimensions of the user–item matrix using the artist transform did not affect predictive accuracy. In an extreme scenario, where hundreds of thousands of items are involved, we suggest that dimensionality reduction would need to use another level of granularity. Were more precise genre description available than in the Smart Radio domain, the user–item matrix could be mapped to a user–genre matrix, greatly reducing the dimensions used when calculating neighbourhoods. However, the accuracy of recommendations made using genre as a basis for correlation, would depend on the accuracy of the genre descriptions in the first place.

In Smart Radio we addressed the problem of recommendations using very little content description. Were more precise content description available, we could improve our case representation and make better context-based recommendations. As it stands, we have demonstrated how even a little content description can improve the effectiveness of the ACF algorithm for users.

8.6.2 Playlist Maintenance

Currently in Smart Radio all adapted playlists are saved for re-use by other users. Since adapted playlists may differ by one or two tracks from the playlist from which they are adapted, an obvious maintenance issue arises as more and more playlists are created. In this context, Smart Radio is a typical example of a long-lived CBR system which accumulates a large amount of case data. There are several maintenance problems for such systems.

One of these is ‘concept drift’, whereby changes to the external domain over time cause cases in the case base to become inaccurate (Salganicoff, 1997). In such situations, the most similar case retrieved is no longer useful for problem solving within the current context. Although this problem has not manifested itself yet, we suggest that because of trends in the popularity of music, the playlist case base in Smart Radio will suffer from this problem. This essentially means that the case representation used in Smart Radio will not be able to capture the emergence of newer, more fashionable artists that listeners may want to hear. Leake and Wilson (1998) suggest an introspective analysis of retrieval activity in the case base to detect whether concept drift is taking place. Racine and Yang (1998) observe that, in a long-lived CBR system, the most recently used cases may be the most useful. Thus, the case base needs to be periodically reorganised to make sure such cases are accessible to the user. Such a policy would appear to implicitly address the problem of concept drift by simply making older cases less accessible. In Smart Radio, implementing such a policy would be problematic because, to perform the first stage retrieval, the system relies on ACF which cannot recommend new items until sufficient users have already endorsed them. However, it is entirely feasible that the introspective reasoning suggested by Leake and Wilson (1998) could be applied to individual tracks in the database to discover those tracks that are becoming increasingly popular within certain segments of the user population. These tracks could be marked so that playlists containing these tracks could be included in the recommendation set. While ACF carries out this policy using rating data submitted by the user population, the difficulty, as we have discussed in this thesis, is that this data reflects *cumulative* rather than emerging interests. An extension to the work carried out in this thesis would be to perform introspective learning on ACF data to isolate localised trends and incorporate this into the recommendation process.

However, we are still left with the issue of an ever increasing amount of playlists, and the resultant utility problems. Rather than by deletion or editing, we suggest that maintenance in Smart Radio could be carried out by introspectively monitoring playlist usage – playlists that are no longer retrieved or are no longer played can be retired from service.

8.6.3 Collaborative Maintenance

One of the aspects of this system is that it requires users to compile a certain number of playlists which are then distributed to other users. Many users are passive users of the system, building very few playlists. Were all users to ‘freeload’, the flow of new recommendations would eventually stop. Therefore, we suggest that users who build playlists regularly, and whose playlists are highly rated by other users are rewarded by being given a more formal role in Smart Radio. Ferrario and Smyth (2001) use the term *collaborative maintenance* to describe their distributed knowledge management framework, in which selected users, *information mediators*, act as moderators, reviewing new material added by other users of the system. This technique is designed to ensure case integrity in a distributed environment where users may post or edit new information. Implementing such architecture is the next obvious step for the Smart Radio system. In such a scenario, information mediators could rate and review new playlists, thus seeding the ACF process. The maintenance issue discussed earlier might also be addressed, were information mediators willing also to validate the retirement of certain playlists.

8.6.4 ACF User Maintenance

A related issue is the idea of user maintenance. In this thesis we have considered the ACF process from the point of view of an interactive, k -nearest neighbour methodology. In such a technique neighbours are assembled using a similarity measure, such as the Pearson coefficient. However, such similarity measures do not take into account attributes that might be valuable in a neighbour in an ACF context. We suggest that a *live* neighbour, one who is still using the system periodically, can be more useful than a neighbour who has not posted data in a long time. Because of data sparsity, ACF systems generally have to accumulate as much data as possible in order to make recommendations. This means that datasets on which recommendations are based contain ‘live user’ data as well as data from users who no longer use the system. While the latter are still valuable in contributing ratings towards the generation of the recommendation set, a neighbourhood assembled completely from this portion of the dataset will contain few current recommendations. Thus, we suggest that in calculating neighbourhoods, it is important to ensure that a portion of the user data is contributed from current users.

As the ACF database grows, it faces the same maintenance issues as a large case base. While the competence measures used in CBR for maintenance may not be appropriate in an ACF context, other metrics might be employed to test whether a profile is useful, such as activity level and the types of items rated. A user who rates only popular items may be less valuable than a user who rates obscure items, and thus contributes ratings in relation to niche interests. In Chapter 4, we described clustering techniques used in ACF. These techniques can be viewed as partitioning the user base according to the interests of its users. An ACF deletion policy could be based on incrementally removing users where prediction accuracy and coverage for the users in each cluster are not significantly affected.

8.6.5 Diversity

The fact that several playlists may be similar leads to a diversity problem whereby a context sensitive recommendation set could be made up of several very similar playlists that have been derived from the same parent. To some extent, the refraction period inhibits this behaviour in the context-boosted recommendation set by removing playlists that contain tracks that have been recently played. A solution would be to employ Smyth and McClave's diversity technique whereby recommendation sets contain playlists that are dissimilar to each other but still maintain similarity to the target playlist (Smyth & McClave 2001). Another would be to index playlists by their parent-child relationships and thus not present closely related playlists in the same recommendation set.

8.6.6 Adaptation

Smart Radio listeners adapt proposed playlists with little help from the system. An enhancement to the playlist editing facility would be to suggest suitable tracks for completing a playlist. Some completion rules could be derived by mining association rules (Agrawal et al. 1996) from existing playlists. Alternatively, a case completion technique could be employed whereby playlists are incrementally retrieved based on the partially completed playlist. The user is then presented with tracks from the most frequently represented artist in the retrieved playlists. This continues until the playlist is completed. The process is similar to the conversational approach of the *Nacodae* system (Aha et al. 2001) which we describe in Chapter 3.

8.7 Summary

In this thesis we have presented a prototype system for the distribution of music on-line. We have used the ACF algorithm as the basis for distributing the playlist expertise of our users, and demonstrated how it can be enhanced to provide more timely recommendations. In such a framework, our users are crucially important because their preferences dictate which artists become popular. This is in contrast to the current models in the music industry whereby blanket marketing campaigns promote larger, more profitable artists. Currently, the industry is moving to create on-line services that duplicate the centralised distribution and marketing strategies that are currently in place. At the same time it is trying to frighten people away from file-sharing networks by pursuing individual file-sharers in court.

We suggest that the file-sharing networks are partly attractive *because* they are outside the control of the media conglomerates which run the music industry. While music consumers develop an empathy or following for a particular artist, this 'brand' loyalty does not extend to Sony or BMG who may market and own the recordings of the artist. It is unclear yet whether the strategies being pursued by the music industry will be successful. We suggest that by making full use of the possibilities of distributed communication, the music industry is in a position to win back consumers it has lost to file-sharing networks. This not only means providing competitive pricing for on-line music services, but also in providing facilities which enable the flow of communication

between listeners so that the term 'popular music' refers, not just to the top one hundred singles, but to the hundreds of thousands of songs available.

References

- Aamodt, A., Plaza, E. (1994) Case Based Reasoning: foundational issues, methodological variations, and system approaches. *AI Communications* 7(1), 39–59.
- Adar, E., Huberman, B. (2000) Free riding on Gnutella. *First Monday* 5(10).
- Agrawal, R., Manilla, H., Srikant, R., Toivonen, H., Verkamo, A.I. (1996) Fast discovery of association rules, in: Fayyad, U., Piatetsky-Shapiro, U.M., Smyth, G., Uthurusamy, R. (Eds.), *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press, pp. 307–328.
- Aha, D.W. (1997a) A proposal for refining case libraries, in: Bergmann, R., Wilke, W. (Eds.) *Proceedings of the Fifth German Workshop on CBR (TR LSA-97-01E)*.
- Aha, D.W. (Ed.) (1997b) Special issue on lazy learning. *Artificial Intelligence Review*, 11, Kluwer Academic Publishers.
- Aha, D. (1998) Reasoning and Learning: The Lazy-Eager Dimension. Invited Keynote Talk at EWCBR 1998, <http://www.aic.nrl.navy.mil/~aha/>
- Aha, D.W., Breslow, L.A. (1997) Refining conversational case libraries, in: *Proceedings of the Second International Conference on Case Based Reasoning*, Providence, RI, Springer.
- Aha, D., Kibler, D., Albert, M. (1991) Instance-based learning algorithms. *Machine Learning* 6, 37–66.
- Aha, D.W., Maney, T., Breslow, L.A. (1998) Supporting Dialogue Inferencing in Conversational Case-Based Reasoning, in: *Proceedings of the Fourth European Workshop on Case-Based Reasoning*, Dublin, Ireland, Springer, pp. 262–273.
- Aha, D., Breslow, L.A., Munoz-Avila, H. (2001) *Conversational Case based Reasoning*. *Applied Intelligence* 14(1), special issue on “Interactive CBR”, Kluwer.
- Allmusic (2002). E-mail correspondence on licensing arrangements for the access to the allmusic.com database.
- Althoff, K.-D., Richter, M.M. (1999) Similarity and Utility in Non-Numerical Domains, *Mathematische Methoden der Wirtschaftswissenschaften*. Physika-Verlag, pp. 403–413. <http://www.cbr-web.org/documents/RichterSimilarity99.pdf>
- Alur, D., Crupi, J., Malks, D. (2001) *Core J2EE Patterns: Best Practices and Design Strategies*. Prentice Hall/Sun Microsystems Press.
- Arcos, J.L., Lopez de Mantaras, R. (1997) Perspectives: A declarative bias mechanism for case retrieval, in: *Proceedings of ICCBR 1997*, LNAI 1266, Springer-Verlag, pp. 279–290.

- Arslan, B., Ricci, F. (2003) Case Based Session Modelling and Personalization in a Travel Advisory System, in: Proceedings of the AH '2002 Workshop on Recommendation and Personalization in eCommerce, Malaga, Spain, May 28, 2002, pp. 60–69.
- Avery, C., Zeckhauser, R. (1997) Recommender Systems for Evaluating Computer Messages, in: Communications of the ACM, 03 1997.
- Baeza-Yates, R., Ribeiro-Neto, B. (1999) Modern Information Retrieval. Addison Wesley.
- 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.
- Balabanovic, M., Shoham, Y. (1997) Fab: Content-based collaborative recommendation. Communications of the ACM 40(3), 66–72.
- Basu, C., Hirsh, H. (1999) Learning user models for recommendation, in: Proceedings of Workshop on Machine Learning for User Modelling at UM'99, <http://www.dfki.de/~bauer/um99-ws/>
- Basu, C., Hirsh, H., Cohen W. (1998) Recommendation as classification: using social and content-based information in recommendation, in: Proceedings of the 15th National Conference on Artificial Intelligence, Madison, WI, pp. 714–720.
- Belkin, N.J., Croft, B. (1992) Information Filtering and Information Retrieval: Two Sides of the Same Coin? Communications of the ACM 35(12).
- Bentley, J.L. (1975) Multidimensional binary search trees used for associative searching. Communications of the ACM 18(9), 509–517.
- Bergmann, R. (1998) The use of taxonomies for representing case features and local similarity measures, in: Gierl, L, Lenz, M (Eds.), 6th German Workshop on CBR.
- Bergmann, R. Stahl, A. (1998) Similarity measures for object-oriented case representations, in: Smyth, B., Cunningham, P. (Eds.), Proceedings of EWCBR 1998, Lecture Notes in Artificial Intelligence, vol. 1488, pp. 25–36.
- Bergmann, R., Muñoz-Avila, H., Veloso, M., Melis, E. (1998) CBR applied to planning, in: Lenz, M., Bartsch-Spörl, B., Burkhard, H.-D., Wess, S. (Eds.), Case-Based Reasoning Technology, From Foundations to Applications, LNAI 1400, Springer-Verlag, pp. 17–50.
- Bergmann, R., Richter, M.M., Schmitt, S., Stahl, A., Vollrath, I. (2001) Utility-oriented matching: a new research direction for case-based reasoning. In: 9th German Workshop on Case-Based Reasoning, GWCBR 2001.
- Berners-Lee, T. (1998) Semantic Web Road map. Available online at <http://www.w3.org/DesignIssues/Semantic.html>.

- Berners-Lee, T., Hendler, J., Lassila, O. (2001) The Semantic Web. Scientific American feature article, May 2001. Available online at <http://www.scientificamerican.com/2001/0501issue/0501berners-lee.html>.
- Berry, M.W., Dumais, S.T., O'Brian, G.W. (1995) Using linear algebra for intelligent information retrieval. *SIAM Review* 37(4), 573–595.
- Billsus, D., Pazzani, M.J. (1998) Learning Collaborative Information Filters, in: Proceedings of AAAI Workshop on Recommender Systems. AAAI Press, pp. 24–28.
- Billsus, D., Pazzani, M.J. (1999) A hybrid user model for news story classification, in: Kay, J. (Ed.), *User Modeling: Proceedings of the Seventh International Conference, UM99*. Springer, pp. 99–108.
- Birch, A., Davidson, S. (2001) Music, the Prototype for the digital economy. OC&C Strategy Consultants report. Available at http://www.occstrategy.com/live/pub_ins_med.htm
- Birch, A., Davidson, S. (2002) Highlights from the Digital marketplace. OC&C Consultants. Available at http://www.occstrategy.com/live/pub_ins_med.htm
- Bonzano, A., Cunningham, P., Smyth, B. (1997) Using introspective learning to improve retrieval in CBR: A case study in air traffic control, in: Proceedings of the 2nd International Conference on Case Based Reasoning (ICCBR-97). Springer-Verlag.
- Bradley, K., Rafter, R., Smyth, B. (2000) Case-based user profiling for content personalisation, in: Proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web-based Systems, Trento, Italy, August 2000.
- Branting, L.K. (2003) Learning feature weights from customer return-set selections. *The Journal of Knowledge and Information Systems (KAIS)*. In press.
- Breese, J.S., Heckerman, D., Kadie, C. (1998) Empirical analysis of predictive algorithms for collaborative filtering, in: Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence, July 1998, pp. 43–52.
- Brickley, D., Guha, R.V. (Eds.) (2000) Resource Description Framework (RDF) Schema Specification 1.0. W3C Candidate Recommendation 27 March 2000. Available online at <http://www.w3.org/TR/rdf-schema/>.
- Bridge, D. (2002) Towards conversational recommender systems: a dialogue grammar approach, in: Aha, D.W. (Ed.), Proceedings of the Workshop in Mixed-Initiative Case-Based Reasoning, Workshop Programme at the Sixth European Conference in Case-Based Reasoning, 2002, pp. 9–22.

- Brin, S., Page, L. (1998) The anatomy of a large-scale hypertextual web search engine, in: Proceedings of the Seventh World Wide Web Conference (WWW7), Brisbane, also in a special issue of the Journal Computer Networks and ISDN Systems 30(1-7).
- Budzik, J., Hammond, K.J., Marlow, C.A., Scheinkman, A. (1998) Anticipating information needs: Every day applications as interfaces to Internet Information sources, in: Proceedings of the 1998 World Conference on the WWW, Internet, and Intranet.
- Burke, R. (1999) Integrating Knowledge based and Collaborative Filtering Recommender Systems, in: Workshop on AI and Electronic Commerce, AAAI 1999.
- Burke, R. (2000) A Case-Based Approach to Collaborative Filtering, in: Proceedings of the EWCBR 2000, LNAI 1898. Springer-Verlag, Berlin, pp. 370-379.
- Burke, R. (2002) Hybrid recommender systems: surveys and experiments. *User Modeling and User-Adapted Interaction* 12(4), 331-370, Kluwer Press.
- Burkhard, H.-D. (1998) Extending some concepts of CBR – foundations of Case Retrieval Nets, in: Lenz, M., Bartsch-Spörl, B., Burkhard, H.-D., Wess, S. (Eds.), *Case Based Reasoning Technology from Foundations to Applications*, LNAI 1400. Springer-Verlag, pp. 17-50.
- Cavicchi, D. (1998) *Tramps Like Us: Music and Meaning Among Springsteen Fans*. Oxford University Press, New York.
- Chauvin, Y., Rumelhart, D. (1995) *BackPropagation: Theory, Architectures, and Applications*. Lawrence Erlbaum Associates, Hillsdale, NJ.
- Cheeseman, P., Kelly, J., Self, M., Stutz, J., Taylor, W., Freeman, D. (1988) Autoclass: a Bayesian classification system, in: *Proceedings of the Fifth International Conference on Machine Learning*, San Mateo, California, 1988, pp. 54-64.
- Chickering, D., Heckerman, D., Meek, C. (1997) A Bayesian approach to learning Bayesian networks with local structure, in: *Proceedings of Thirteenth Conference on Uncertainty in Artificial Intelligence*, Providence, RI. Morgan Kaufmann.
- Claypool, M., Gokhale, A., Miranda, T. (1999) Combining content-based and collaborative filters in an online newspaper, in: *Proceedings of the ACM SIGIR Workshop on Recommender Systems*.
- Claypool, M., Le, P., Waseda, M., Brown, D. (2001) Implicit interest indicators, in: *Proceedings of ACM Intelligent User Interfaces Conference (IUI)*, Santa Fe, New Mexico, 2001, ACM.
- Clerkin, P., Cunningham P., Hayes C. (2001a) Ontology discovery for the semantic web using hierarchical clustering, in: *Proceedings of Semantic Web Mining Workshop at ECML/PKDD-2001*, September 3, Freiburg, Germany.

- Clerkin, P., Hayes, C., Cunningham, P. (2001b) Automated case generation for recommender systems using knowledge discovery techniques, in: Proceedings of Workshop on Case Based Reasoning in Electronic Commerce at the Fourth International Conference on Case-Based Reasoning 2001, Vancouver BC, Canada.
- Clerkin, P., Cunningham, P., Hayes, C. (2003) Concept discovery in collaborative recommender systems, in: Proceedings of the 14th Irish Conference of Artificial Intelligence and Cognitive Science (AICS 2003), Trinity College Dublin, September 2003.
- Collins, A.M., Loftus, E.F. (1988) A spreading activation theory of semantic processing, in: Coolins, A.M., Smithchapter, E.E. (Eds.), Readings in Cognitive Science. Morgan Kaufman, pp. 126–136, chapter 2.3.
- Condliff, M.K., Lewis, D.D., Madigan, D., Posse, C. (1999) Bayesian Mixed Effect Models for Recommender Systems, in: ACM SIGIR 1999 Workshop in Recommender Systems: Algorithms and Evaluation, University of California, Berkeley August 19.
- Cook, N. (1998) Music: A Very Short Introduction. Oxford University Press.
- Cook, P.R. (Ed.) (1999). Music, Cognition and Computerized Sound: An Introduction to Psychoacoustics. MIT Press, Cambridge, MA.
- Cook, P.R. (2001) Music, Cognition, and Computerized Sound: An Introduction to Psychoacoustics. MIT Press.
- Coyle, L., Cunningham, P., Hayes, C. (2002) Representing cases for CBR in XML, in: Proceedings of 7th UKCBR Workshop, Peterhouse, Cambridge, UK.
- Coyle L, Cunningham P., Hayes C. (2000) A Case-Based Personal Travel Assistant for Elaborating User Requirements and Assessing Offers (2002), in proceedings of the 6th European Conference on Case-Based Reasoning, (2002), Aberdeen, Scotland.
- Cunningham, P. (1998) CBR: strengths and weaknesses, in: del Pobil, A.P., Mira, J., Ali, M. (Eds.), Proceedings of 11th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems. Springer, LNAI 1416, Vol. 2, pp. 517–523.
- Cunningham, P., Bonzano, A. (1999) Knowledge engineering issues in developing a case-based reasoning application. Knowledge Based Systems 12, 372–379.
- Cunningham, P., Smyth, B. (1994) A comparison of model based and incremental case-based approaches to electronic fault diagnosis, in: Proceedings of the Case Based Workshop, AAAI-1994.

- Cunningham, P., Finn, D., Slattery, S., (1994) Knowledge engineering requirements in derivational analogy, in: Wess, S., Althoff, K.-D., Richter, M.M. (Eds.), *Topics in Case-Based Reasoning*, LNAI. Springer Verlag, pp. 234–245.
- Cunningham, P., Smyth, B., Bonzano, A. (1998) An incremental retrieval mechanism for case-based electronic fault diagnosis. *Knowledge-Based Systems* 11(3–4), 239–248.
- Cutler, M., Sterne, J. (2000) E-metrics, business metrics for the new economy. Technical report, NetGenesis Corp., http://www.spss.com/netgenesis/emetrics_reading.htm
- Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R. (1990) Indexing by latent semantic analysis. *Journal of the American Society for Information Science* 41(6), 391–407.
- Delgado, J., Ishii, N. (1999a) Memory-based weighted majority prediction for recommender systems, in: *ACM SIGIR'99 Workshop on Recommender Systems*.
- Delgado, J., Ishii N. (1999b) Online learning of user preferences in recommender systems, in: *Proceedings of the IJCAI-99 Workshop on Machine Learning for Information Filtering*.
- Delgado, J., Ishii N., Ura, T. (1998) Content-based collaborative information filtering, in: *Cooperative Information Agents II, Lecture Notes in Artificial Intelligence 1435*. Springer Verlag, Berlin, Heidelberg, New York, pp. 206–215.
- Dempster, A.P., Laird, N.M., Rubin, D.B. (1977) Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society Series B* 39(1), 1–38.
- Denning, P. (1982) Electronic junk. *Communications of the ACM* 23(3), 163–165.
- Desain, P., Honing, H., van Thienen, H., Windsor, L. (1998) Computational Modeling of Music Cognition: Problem or Solution? *Music Perception* 16 (1), 151–166.
- Doyle, M., Cunningham, P. (2000) A dynamic approach to reducing dialog in on-line decision guides, in: Blanzieri, E., Portinale, L. (Eds.), *Proceedings of EWCBR 2000*, LNAI 1898. Springer Verlag, pp. 49–60.
- Durlacher Research Ltd. (2001) *Impacts of Digital Distribution on the Music Industry*. Author: Jay Marathe.
- Fayyad, U., Piatetsky-Shapiro, G., Smyth, P. (1996) From data mining to knowledge discovery in databases. *AI Magazine* 17(3), 37–54.
- Feiten, B., Gunzel, S., (1994) Automatic indexing of a sound database using self-organizing neural nets. *Computer Music Journal* 18(3) Summer.
- Feld, S. (1994) Communication, music and speech about music, in: Keil, C., Feld, S. (Eds.) *Music Grooves: Essays and Dialogues*. University of Chicago Press.

- Fernback, J. (2003) Legends on the Net. *New Media and Society* 5(1) Jankowski, N., Jones, S., Lievrouw, L., Silverstone, R. (Eds.).
- Fernback, J., Thompson, B., (1995) Virtual Communities: Abort, Retry, Failure. Originally presented as: Computer-Mediated Communication and the American Collectivity: The Dimensions of Community Within Cyberspace, in: Annual Convention of the International Communication Association, Albuquerque, New Mexico, May 1995. Available at <http://www.well.com/user/hlr/texts/VCCivil.html>
- Ferrario, M.A., Smyth, B. (2000) Collaborative maintenance – a distributed, interactive case-based maintenance strategy, in: Proceedings of 5th European Workshop on Case Based Reasoning, Trento, Italy.
- Ferrario, M.-A., Smyth, B. (2001) Distributing case-base maintenance, the collaborative maintenance approach. *Journal of Computational Intelligence: Special Issue on Maintaining Case-Based Reasoning Systems* 17(2), 315–330.
- Fisher, D.H. (1987) Knowledge acquisition via incremental conceptual clustering. *Machine Learning* 2, 139–172.
- Fisher, W. (2000) Digital Music: Problems and Possibilities. Available at <http://www.law.harvard.edu/faculty/tfisher/Music.html>
- Fisher, W. (2003) Technology, Law, and the Future of Entertainment. Draft copy obtained from author. Faculty of Law, Harvard University.
- Foltz, P.W., Dumais, S.T. (1992) Personalized information delivery: An analysis of information filtering methods. *Communications of the ACM* 35(12), 51–60.
- Foote, J. (1999) An overview of Audio information retrieval. *Multimedia Systems*. Springer-Verlag, 7, 2–10.
- Fox, S., Leake, D. (1995) Learning to refine indexing by introspective reasoning, in: Proceedings of First International Conference on Case-Based Reasoning, ICCBR '95, Sesimbra, Portugal. Springer Verlag.
- Friedman, J., Bentley J., Finkel, R. (1977) An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software* 3(3), 209–226.
- Frith, S. (1988) *Why do Songs have Words in Music for Pleasure*. Routledge, New York.
- Frith, S. (1993) *Music and Copyright*. Edinburgh University Press.
- Gang, D., Goldman, C.V., Lehmann, D., Rosenschein, J.S. (1999) NetNeg: a connectionist-agent integrated system for representing musical knowledge. *Annals of Mathematics and Artificial Intelligence* 25(1/2), 69–90, Kluwer Academic Publishers.

- Gebhardt, F. (1997) Survey on structure-based case retrieval. *The Knowledge Engineering Review* 12, 41–58.
- Gentner, D. (1983) Structure mapping: a theoretical framework for analogy. *Cognitive Science* 7, 155–170.
- Gentner, D., 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.
- Goker, A. (1999) Capturing information need by learning user context, in: *Sixteenth International Joint Conference in Artificial Intelligence: Learning About Users Workshop*, pp. 21–27.
- Goker, M.H. (2002) Introduction to Workshop on Case Based Reasoning and Personalization, in: Goker, M.H., Smyth, B. (Eds.), *Workshop on Case Based Reasoning and Personalization, ECCBR 2002, Aberdeen, Scotland*.
- Goker, A., McCluskey, T.L. (1991) *Toward an adaptive Information retrieval system*
- Gokhale, A., Claypool, M. (1999) Thresholds for more accurate collaborative filtering, in: *Proceedings of the IASTED International Conference Artificial Intelligence and Soft Computing, August 9–12, 1999, Honolulu, Hawaii, USA*.
- Goldberg, A., Robson D. (1983) *Smalltalk-80: The Language and its Implementation*. Addison-Wesley Publishers, Menlo Park.
- Goldberg, D., Nichols, D., Okri, B.M., Terry, D. (1992) Using collaborative filtering to weave an information tapestry. *Communications of the ACM* 35(12) 61–70.
- Golle, P., Leyton-Brown, K. (2001) Incentives for sharing in peer-to-peer networks, in: *Proceedings of the Third ACM Conference on Electronic Commerce, Tampa, Florida, USA, October 2001*.
- Griffith, D. (2003) From lyric to anti-lyric: analysing the words in a pop song, in: Moore, A.F. (Ed.), *Analysing Popular Music*. Cambridge University Press.
- Grimaldi, M., Cunningham, P., Kokaram, A. (2003) An evaluation of alternative feature selection strategies and ensemble techniques for classifying music. To appear in: *Proceedings of the ECML2003 Workshop on Multimedia Cavtat, Croatia, September 22–26, 2003*.
- Grudin, J. (1994) Groupware and social dynamics: eight challenges for developers. *Communications of the ACM* 37, 92–105.
- Guralnick, P. (1994) *Last Train to Memphis: The Rise of Elvis Presley*. Little, Brown, Boston.
- Harrison, A. (2001) *Music – The Business: The Essential Guide to the Law and the Deals*. Virgin Publishing, London.

- Hart, P.E. (1968) The condensed nearest neighbor rule. *IEEE Transactions on Information Theory* 14(3) May.
- Hayes, C. (1999) Smart Radio – Building Smart Music Radio, in *Proceedings of AICS'99, Irish Conference of Cognitive Science and Artificial Intelligence*, University College Cork, Ireland.
- Hayes, C. (1999) Smart Radio – an Application Proposal. Technical report, Department of Computer Science, Trinity College Dublin Available at <http://www.cs.tcd.ie/publications/tech-reports/reports.99/TCD-CS-1999-24.pdf>
- Hayes, C., Cunningham, P. (2000) Smart Radio – building community based music radio, in: Macintosh, A., Moulton, M., Coenen, F. (Eds.), *Applications and Innovations in Intelligent Systems VIII*, BCS Conference Series. Springer-Verlag.
- Hayes, C., Cunningham, P. (2001) SmartRadio – community based music radio. *Knowledge Based Systems*, special issue ES2000 14(3–4) June, Elsevier.
- Hayes C., Cunningham P., Doyle M. (1998) Distributed CBR using XML, in: *Proceedings of the KI-98 Workshop on Intelligent Systems and Electronic Commerce*, number LSA-98-03E. University of Kaiserslauten Computer Science Department, 1998. Also available as TCD technical report TCD-CS-1998-06 <http://www.cs.tcd.ie/publications/tech-reports/tr-index.98.html>
- Hayes, C., Cunningham, P., Smyth, B. (2001) A case-based reasoning view of automated collaborative filtering, in: Aha, D.W., Watson, I. (Eds.), *Proceedings of 4th International Conference on Case-Based Reasoning*, LNAI 2080. Springer Verlag, pp. 234–248.
- Hayes, C., Cunningham, P., Clerkin, P., Grimaldi, M. (2002a) Programme driven radio, in: van Harmelen (Ed.), *Proceedings of ECCAI 2002, 15th European Conference on Artificial Intelligence 2002*, Lyon, France, IOS Press.
- Hayes, C., Massa, P., Avesani, P., Cunningham, P. (2002b). An on-line evaluation framework for recommender systems, in: *Proceedings of the IWorkshop on Recommendation and Personalization Systems*, AH 2002, Malaga, Spain, 2002. Springer-Verlag.
- Hebdige, D. (1981) *Subculture, the Meaning of Style*, New Accents series. Routledge Press, London.
- Heckerman, D., Geiger, D., Chickering, D. (1995) *Learning Bayesian Networks: The combination of knowledge and statistical data*. Machine Learning 20, 197, Kluwer Academic Publishers.
- Herlocker, J., Konstan, J., Borchers, A., Riedl, J. (1999) An Algorithmic framework for performing collaborative filtering, in: *Proceedings of ACM SIGIR'99*. ACM Press.
- Herlocker, J.L., Konstan, J.A., Riedl, J. (2000) Explaining collaborative filtering recommendations, in: *Proceedings of ACM 2000 Computer Supported Cooperative Work (CSCW)*, pp. 241–250.

- Hill, W.C, Hollan, J.D., Wroblewski, D., McCandless, T. (1992) Using collaborative filtering to weave an information tapestry. *Communications of the ACM* 35(12), 61–70, December.
- Hill, W., Stead, L., Rosenstein, M., Furnas, G. (1995) Recommending and evaluating choices in a virtual community of use, in: *Proceedings of the Conference on Human Factors in Computing Systems, CHI'95*. <http://www.acm.org/sigchi/chi95/>
- Hiltz, S.R., Turoff, M. (1985) Structuring computer-mediated communication systems to avoid information overload. *Communications of the ACM* 28(7), 680–689.
- Hofinaun, Y., Puzicha, J. (1999) Latent class models collaborative filtering, in: *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pp. 688–693.
- Jain, A.K., Dubes, R.C. (1988) *Algorithms for Clustering Data*. Prentice Hall.
- Kamp, G., Pirk, P., Burkhard, H.-D. (1996) Falldaten case-based reasoning for the diagnosis of technical devices, in: Görz, G., Hölldobler, S. (Eds.), *Proc. KI-96, Advances in Artificial Intelligence, Lecture Notes in Artificial Intelligence 1137*, Dresden, Germany, September 1996. Springer Verlag.
- Keil, C. (1994) Motion and feeling through music, in: Keil, C., Feld, S. (Eds.), *Music Grooves: Essays and Dialogues*. University of Chicago Press.
- Kerman, J. (1986) *Contemplating Music: Challenges to Musicology*. Harvard University Press.
- Kohrs, A., Merialdo, B. (1999) Clustering for collaborative filtering applications, in: *Computational Intelligence for Modelling, Control & Automation*. IOS Press.
- Kolodner, J.L. (1993) *Case Based Reasoning*. Morgan Kaufmann, San Mateo.
- Konstan, J.A., Riedl, J. (1999) Research resources for recommender systems, in: *CHI'99 Workshop Interacting with Recommender Systems*, <http://www.darmstadt.gmd.de/rec99/>
- Konstan, J.A., Miller, B., Maltz, D., Herlocker, J., Gordon, L., Riedl, J. (1997) GroupLens: applying collaborative filtering to Usenet news. *Communications of the ACM* 40(3), 77–87.
- Lang, K. (1995) NewsWeeder: Learning to filter netnews, in: *Machine Learning: Proceedings of the Twelfth International Conference, Lake Tahoe, California*.
- Leach, E., Henslee, B. (2001) Follow the money: who's really making the dough? in: *Electronic Musician*, Nov 1, PRIMEDIA Business Magazines & Media Inc.
- Leake, D. (Ed.) (1996) *Case-Based Reasoning: Experiences, Lessons, and Future Directions*. AAAI Press/MIT Press, Menlo Park.
- Leake, D., Wilson, D. (1998) Categorizing case-base maintenance: dimensions and directions, in: Smyth, B. Cunningham, P. (Eds.), *Advances in Case-Based Reasoning: Proceedings of the*

- Fourth European Workshop on CaseBased Reasoning, Berlin, Germany, September 1998. Springer-Verlag, pp. 196–207.
- Leake, D., Wilson, D. (1999) When experience is wrong: Examining CBR for changing tasks and environments, in: Proceedings of the Third International Conference on Case-Based Reasoning. Springer-Verlag, Berlin, 218–232.
- Lenz, M. (1993) Cabata – a hybrid cbr system, in: Althoff, K.-D., Richter, K., Wess, S. (Eds.), Proceedings of the First European Workshop on Case-Based Reasoning, Kaiserslautern, pp. 204–209.
- Lenz, M. (1996) Preparing case retrieval nets for distributed processing, in: Czaja, L., Starke, P., Burkhard, H.-D., Lenz, M. (Eds.), Proceedings of the Workshop on Concurrency, Specification and Programming (CS&P-96), Humboldt University, Berlin, 1996.
- Lenz, M. (1999) Case Retrieval Nets as a model for building flexible information systems. PhD dissertation, Faculty of Mathematics and Natural Sciences, Humboldt University, Berlin.
- Lenz, M., Burkhard, H.-D. (1996) Case Retrieval Nets: Basic ideas and extensions, in: Gorz, G., Holldobler, S. (Eds.), KI-96: Advances in Artificial Intelligence, Lecture Notes in Artificial Intelligence 1137. Springer Verlag, pp. 227–239.
- Lenz, M., Auriol, E., Manago, M. (1998) Diagnosis and decision support, in: Lenz, M., Bartsch-Spörl, B., Burkhard, H.-D., Wess, S. (Eds.), Case Based Reasoning Technology from foundations to applications, LNAI 1400. Springer-Verlag, pp. 17–50.
- Lenz, M., Hubner, A., Kunze, M. (1998) Question answering with textual CBR, in: Andreasen, T., Christiansen, H., Larsen, H.L. (Eds.), Flexible Query Answering Systems, Lecture Notes in Artificial Intelligence 1495. Springer Verlag.
- Lieberman, H. (1997) Letizia: an agent that assists web browsing, in: Proceedings of the International Joint Conference on Artificial Intelligence IJCAI-95, Montreal, 1995.
- Lieberman, H., Fry, C., Weitzman, L. (2001) Exploring the web with reconnaissance agents. Communications of the ACM 44(8) August.
- Liebowitz, S. (2003) Will MP3 downloads annihilate the record industry? The evidence so far, in: Libecap, G. (Ed.), Advances in the Study of Entrepreneurship, Innovation and Economic Growth. JAI Press.
- Loeb, S. (1992) Architecting personalized delivery of multimedia information. Communications of the ACM 35(12) December.
- Maes, P. (1994) Agents that reduce work and information overload. Communications of the ACM 37(7), 31–40, July.

- Malone, T.W., Grant, K.R., Turbak, F.A., Brobst, S.A., Cohen, M.D. (1987) Intelligent information sharing systems. *Communications of the ACM* 30(5), 390–402, May.
- Maltz, D., Ehrlich, K. (1995) Pointing the way: Active Collaborative Filtering, in: *Proceedings of ACM SIGCHI Conference, 1995*.
- Manago, M., Bergmann, R. (1994) CASUEL: A Common Case Representation Language. <http://wwwagr.informatik.uni-kl.de/~bergmann/casuel/>
- Mann, C. (2000) The Heavenly Jukebox. *The Atlantic Monthly Magazine*, September 2000 <http://www.theatlantic.com/issues/2000/09/mann.htm>
- McCulloch, W.S., Pitts, W. (1943) A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics* 5, 115–133.
- McSherry, D. (2001) Interactive case based reasoning in sequential diagnosis. *Applied Intelligence* 14, 65–76.
- Meyer, L. (1956) *Emotion and Meaning in Music*. University of Chicago Press.
- Miller, B., Riedl, J., Konstan, J. (1997) Experiences with GroupLens: making Usenet useful again, in: *Proceedings of the USENIX 1997 Annual Technical Conference, Anaheim, CA*, pp. 219–231.
- Minsky, M.L. (1961) Steps toward artificial intelligence, in: *Proceedings of the Institute of Radio Engineers* 49, pp. 8–30.
- Mitchell, T. (1997) *Machine Learning*. McGraw Hill.
- Miyahara, K., Pazzani, M.J. (2000) Collaborative filtering with the simple Bayesian classifier, in: *Proceedings of the Pacific Rim International Conference on Artificial Intelligence*, pp. 679–689.
- MMF (2001) *The MMF Guide to Professional Music Management*. Sanctuary, London. ISBN 1860743552.
- Mobasher, B., Honghua, D., Tao, L., Miki, N. (2000a) Improving the Effectiveness of Collaborative Filtering on Anonymous Web Usage Data. Technical Report. DePaul University, Chicago, Illinois, School of Computer Science, Telecommunication and Information Systems.
- Mobasher, B., Honghua, D., Tao, L., Miki, N., Witshire, J. (2000b) Discovery of aggregate usage profiles for web personalization, in: *Proceedings of the WebKDD Workshop at the ACM SIGKDD, Boston, August 2000*.
- Mobasher, B., Berendt, B., Spiliopoulou, M. (2001) KDD for Personalization – PKDD 2001 Tutorial, September 6, 2001, KDD.

- Moore, A.W., Lee, M.S. (1994) Efficient algorithms for minimising cross validation error, in: Proceedings of the 11th International Conference on Machine Learning. Morgan Kaufmann, San Francisco.
- Morita, M., Shinoda, Y. (1994) Information filtering based on user-behaviour analysis and best match text retrieval, in: Proceedings of SIGIR '94, ACM, New York.
- Myaeng, S.H., Korfhage, R.R. (1986) Towards an intelligent and personalized retrieval system, in: Proceedings of the ACM SIGART International Symposium on Methodologies for Intelligent Systems, Knoxville, Tennessee, USA. ACM Press, pp. 121–129.
- Nakamura, A., Abe., N. (1998) Collaborative filtering using weighted majority prediction algorithms, in: Machine Learning: Proceedings of the Fifteenth International Conference, 1998.
- Nebel, B., Koehler, J. (1995) Plan reuse versus plan generation: a theoretical and empirical analysis. *Artificial Intelligence (Special Issue on Planning and Scheduling)* 76(1–2), 427–454.
- Newell, A., Simon, H. (1976) Computer science as empirical inquiry: symbols and search. *Communications of the ACM* March, 113–126.
- Nichols, D.M. (1997) Implicit rating and filtering, in: Proceedings of the Fifth DELOS Workshop on Filtering and Collaborative Filtering, November 1997.
- O'Connor, M., Herlocker, J. (1999) Clustering items for collaborative filtering, in: Proceedings of ACM SIGIR 1999 Workshop on Recommender Systems: Algorithms and Evaluation, University of California, Berkeley, August 19, 1999.
- Oard, D., Kim, J. (1998) Implicit feedback for recommender systems, in: Proceedings of the AAAI Workshop on Recommender Systems, July 1998.
- Oram, A. (Ed.) (2001) *Peer-to-Peer, Harnessing the Power of Disruptive Technologies*. O'Reilly & Associates.
- O'Sullivan, D., Wilson, D., Smyth, B. (2002) Improving collaborative personalized TV services: a study of implicit and explicit user profiling, in: Proceedings of the 22nd SGAI International Conference on Knowledge Based Systems and Applied Artificial Intelligence, Cambridge, England.
- Palme, J. (1984) You have 134 unread mail do you want to read them now? in: IFIP Conference on Computer Based Message Services, Nottingham, UK, IFIP.
- Pater, W. *The School of Giorgione, Studies in the History of the Renaissance*.
- Pavel, T. (1989) *The Feud of Language: a History of Structuralist Thought*. Blackwell, Oxford.
- Pazzani, M.J. (1999) A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review* 13(5/6), 393–408.

- Pazzani, M., Muramatsu, J., Billsus, D. (1996) Syskill & Webert: Identifying interesting Web Sites, in: Proceedings of the Thirteenth National Conference on Artificial Intelligence, Portland, 1996, pp. 54–61.
- Pennock, D.M., Horvitz, E., Giles, C.L. (2000) Social choice theory and recommender systems: Analysis of the axiomatic foundations of collaborative filtering, in: Proceedings of the Seventeenth National Conference on Artificial Intelligence, 2000, pp. 729–734.
- Plomp, R. (1976) Aspects of Tone Sensation: A Psychophysical Study. Academic Press, London.
- Popescul, A., Ungar, L., Pennock, D.M., Lawrence, S. Probabilistic models for unified collaborative and content based recommendation in sparse data environments, in: Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence.
- Quinlan, J.R. (1993) C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo, CA.
- Quinlan, J.R. (1986). Induction of Decision Trees, Machine Learning, (1), 81-106
- Racine, K., Yang, Q. (1997) Maintaining unstructured case bases. Case-based reasoning research and development, in: Proceedings of the 2nd International Conference on Case-Based Reasoning, Providence, RI, USA, pp. 553–564.
- Rafter, R., Smyth, B. (2001) Passive profiling from server logs in an online recruitment environment, in: Proceedings of the IJCAI Workshop on Intelligent Techniques for Web Personalisation (ITWP 2001), Seattle, Washington, USA.
- Rafter, R., Bradley, K., Smyth, B. (1999) Passive profiling and collaborative recommendation, in: Proceedings of the 10th Irish Conference on Artificial Intelligence and Cognitive Science, Cork, Ireland, September 1999.
- Rafter, R., Smyth, B., Bradley, K. (2000) Inferring relevance feedback from Server Logs: A Case Study in Online recruitment, in: Proceedings of the 11th Irish Conference on Artificial Intelligence and Cognitive Science (AICS 2000), Galway, Ireland.
- Resnick, P., Varian, H. R. (1997) Recommender Systems. Communications of the ACM 40(3), 56–58.
- Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J. (1994) An open architecture for collaborative filtering of Netnews, in: ACM Conference on Computer Supported Co-operative Work, 1994, pp. 175–186.
- Rich, E. (1989) Stereotypes and user modeling, in: Kobsa, A., Wahlster, W. (Eds.), User Models in Dialog Systems, Springer-Verlag, Berlin, pp. 35–51.
- Richter, M. (1995) The knowledge contained in similarity measures. Invited talk at ICCBR95, <http://www.cbr-web.org/documents/Richtericcbr95remarks.html>

- Richter, M.M. (1998) Introduction (to Case-Based Reasoning), in: Lenz, M., Bartsch-Spörl, B., Burkhard, H.-D., Wess, S. (Eds.), *Case-Based Reasoning Technology: from Foundations to Applications*. Springer-Verlag, LNAI 1400, pp. 1–16.
- Riesbeck, C.K., Schank, R.C. (1989) *Inside Case-Based Reasoning*. Lawrence Erlbaum Associates, Cambridge, MA.
- Rocchio, J.J. (1971) Relevance feedback in information retrieval, in: Salton, G. (Ed.), *The Smart Retrieval System – Experiments in Automatic Document Processing*. Prentice Hall, Englewood Cliffs, NJ, pp. 313–323.
- Rose, T. (1994) *Black Noise*. University Press of New England, Hanover, NH.
- Salganicoff, M. (1997). Tolerating concept and sampling shift in lazy learning using prediction error context switching. *Artificial Intelligence Review*, 11(1-5):133-155, 1997.
- Salton, G., Buckley, C. (1990) Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science* 41(4), 288–297.
- Salton, G., Buckley, C. (1998) Term-weighting approaches in automatic text retrieval. *Information Processing and Management* 24(5), 513–523.
- Salton, G., Wong, A., Yang, C.S. (1975) A vector space model for automatic indexing. *Communications of the ACM* 18(11) November.
- Sanders, K.E., Kettler, B.P., Hendler, J.A. (1997) The case for graph-structured representations, *ICCBR 97*, Springer-Verlag, Berlin, pp. 245–254.
- Saroiu, S., Gummadi, P., Gribble., S. (2001) Measurement study of peer-to-peer file sharing systems. Tech Report UW-CSE-01-06-02, University of Washington.
- Sarwar, B., Konstan, J., Borchers, A., Herlocker, J., Miller, B., Riedl, J. (1998) Using filtering agents to improve prediction quality in the GroupLens research collaborative filtering system, in: *Proceedings of ACM CSCW, 1998*, Seattle, Washington.
- Sarwar, B., Karypis, G., Konstan, J., Riedl, J. (2000a) Application of dimensionality reduction in recommender systems – a case study, in: *ACM WebKDD Workshop, 2000*.
- Sarwar, B., Karypis, G., Konstan, J., Riedl, J. (2000b) Analysis of recommendation algorithms for e-commerce, in: *Proceedings of ACM E-Commerce, 2000*.
- Sarwar, B., Karypis, G., Konstan, J., Riedl, J. (2001) Item-based collaborative filtering recommendation algorithms, in: *Proceedings of the 10th International World Wide Web Conference*, pp. 285–295.
- Schafer, J.B., Konstan, J., Riedl, J. (1999) Recommender systems in E-Commerce, in: *EC'99: Proceedings of the First ACM Conference on Electronic Commerce*, Denver, CO, pp. 158–166.

- Schank, R.C. (1982) *Dynamic Memory: A Theory of Learning in Computers and People*. Cambridge University Press, New York.
- Schank, R., Abelson R. (1977) *Scripts, Plans, Goals and Understanding*. Lawrence Erlbaum Associates, Hillsdale, NJ.
- Schein, A.I, Popescul, A., Ungar, L. (2002) Methods and metrics for cold start recommendations, in: *Proceedings of ACM SIGIR 2002*, Tampere, Finland.
- Schlit, B., Adams, N., Want, R. (1994) Context-aware computing applications, in: *Proceedings Workshop on Mobile Computing Systems and Applications*, IEEE, December 1994.
- Schmitt, S., Bergmann, R. (2001) *A Formal Approach to Dialogs With Online Customers*
- Schumacher, J., Bergmann, R. (2000) An effective approach for similarity-based retrieval on top of relational databases, in: *Fifth European Workshop on Case-Based Reasoning*. Springer-Verlag.
- Shardanand, U., Maes, P. (1995) Social information filtering: Algorithms for automating “word of mouth”, in: *Human Factors in Computing Systems CHI '95 Conference Proceedings*, pp. 210–217.
- Shepherd, J. (1991) *Music as Social Text*. Cambridge, UK.
- Sheth, B., Maes, P. (1993) Evolving agents for personalized information filtering, in: *Proceedings of the 9th Conference on Artificial Intelligence for Applications (CAIA'93)*, Orlando, FL, March 1993. IEEE Computer Society Press, pp. 345–352.
- Shimazu, H., Kitano, H., Shibata, A. (1993) Retrieving cases from relational databases: Another strike toward corporate-wide case-based systems, in: *Proceedings of the 13th International Joint Conference in Artificial Intelligence (IJCAI'93)*.
- Slater, D. (1997) *Consumer Culture and Modernity*. Polity Press, Cambridge.
- Sloboda, J.A. (1985) *The Musical Mind: The Cognitive Psychology of Music*. Oxford University Press, Oxford.
- Smoliar, S.W., Wilcox, L.D. (1997) Indexing the content of multimedia documents, in: *Proceedings: VISual'97; Second International Conference on Visual Information Systems*, San Diego, CA, pp. 53–60.
- Smoliar, S.W., Baker, J.D., Nakayama, T., Wilcox, L. (1996) Multimedia search: An authoring perspective, in: *Proceedings of the First International Workshop on Image Databases and Multimedia Search*, IAPR, August 1996, pp. 1–8.
- Smyth, B., Cotter, P. (1999a) The sky's the limit: a personalised TV listings service for the digital TV age, in: *Proceedings of 19th SGAI International Conference on KBS and Applied AI (ES)*, Cambridge, UK.

- Smyth, B., Cotter, P. (1999b) Surfing the Digital Wave: generating personalised TV listings using collaborative, case-based recommendation, in: Althoff, K.-D., Bergmann, R., Branting, L.K. (Eds.) Proceedings of ICCBR 1999, LNAI 1650. Springer Verlag, pp. 561–571.
- Smyth, B., Cunningham, P. (1996) The utility problem analysed, a case base reasoning perspective, in: Smith, I., Faltings, B. (Eds.), EWCBR'96 Advances in Case-Based Reasoning, Lecture Notes in Artificial Intelligence. Springer-Verlag, pp. 392–399.
- Smyth, B., Keane, M. (1995) Remembering to forget: A competence-preserving case deletion policy for case-based reasoning systems, in: Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence. Morgan Kaufmann, San Francisco, pp. 377–382.
- Smyth, B., Keane, M.T. (1998) Adaptation-guided retrieval: questioning the similarity assumption in reasoning. *Artificial Intelligence* (102)2, 49–293.
- Smyth, B., McClave, P. (2001) Similarity vs. diversity, in: Proceedings of the 4th International Conference on Case-Based Reasoning, Vancouver, Canada, 2001. Springer-Verlag.
- Smyth, B., McKenna, E. (1998) Modeling the competence of case-bases, in: Smyth, B., Cunningham, P. (Eds.), Advances in Case-Based Reasoning: Proceedings of EWCBR 1998, LNAI 1488. Springer-Verlag, Berlin, Germany, September, 1998, pp. 196–207.
- Smyth, B., McKenna, E. (1999) Case-based reasoning research and development, in: Althoff, K.-D., Bergmann, R., Branting, L.K. (Eds.), Lecture Notes in Artificial Intelligence. Springer-Verlag, pp. 343–357.
- Smyth, B., Cotter, P., O'Hare, G. (1998) Let's get personal: personalised TV listings on the web, in: Proceedings of Irish Conference of Artificial Intelligence and Cognitive Science, AICS 1998.
- SoundFisher (2001) SoundFisher 1.0 User's Guide. Available at <http://www.soundfisher.com/manual/index.html> (August 2003).
- Stahl, A. (2001) Learning feature weights from case order feedback, in: Proceedings of the 4th International Conference on Case-Based Reasoning, ICCBR 2001.
- Stahl, A. (2002) Defining similarity measures: top-down vs. bottom-up, in: Craw, S., Preece, A. (Eds.) Proceedings of ECCBR 2002, Aberdeen, Scotland, UK, September 2002, LNAI 2416. Springer-Verlag, Berlin.
- Stefik, M. (1997a) Letting loose the light: igniting commerce in electronic publication, available in: *Internet Dreams: Archetypes, Myths, and Metaphors*. MIT Press, Cambridge, MA.
- Stefik, M. (1997b) Shifting the possible: how trusted systems and digital property rights challenge us to rethink digital publishing. *Berkeley Law Journal* 12, 137. <http://www.law.berkeley.edu/journals/btlj/articles/vol12/Stefik/html/reader.html>

- Swearingen, K., Sinha, R. (2001) Beyond algorithms: An HCI perspective on recommender systems, in: ACM SIGIR 2001 Workshop on Recommender Systems, New Orleans, Louisiana, 2001.
- Sweeny, J. (1973) An experimental investigation of the free-rider problem. *Social Science Research* 2.
- Turing, A.M. (1950) Computing machinery and intelligence. *Mind* 59, 433–460.
- Tzanetakis, G., Essl, G., Cook, P. (2000) Automatic music genre classification of audio signals, in: *Proceedings of the International Symposium on Music Information Retrieval (ISMIR)*, Bloomington, Indiana, 2000.
- Ungar, L.H., Foster, D.P (1998a) Clustering methods for collaborative filtering, in: *Proceedings of the Workshop on Recommendation Systems*. AAAI Press, Menlo Park, California.
- Ungar, L.H., Foster, D.P. (1998b) A formal statistical approach to collaborative filtering, in: *Proceedings of CONALD'98*.
- Waszkiewicz, P., Cunningham, P., Byrne, C. (1999) Case-based user profiling in a personal travel assistant, user modeling, in: Kay, J. (Ed.), *Proceedings of the 7th International Conference, UM99*. Springer, Wien, New York, pp. 323–325.
- Weibelzahl, S. (2001) Framework for the evaluation of adaptive CBR systems, in: *Proceedings of the 9th German Workshop on Case-Based Reasoning, Baden-Baden, Germany 2001*, pp. 254–263.
- Wess, S., Althoff, K.-D., Derwand, G. (1993) Using kd-trees to improve the retrieval step in case-based reasoning, in: Wess, S., Althoff, K.-D., Richter, M.M. (Eds.), *Topics in Case-Based Reasoning*, number 837 in *Lecture Notes in AI*. Springer-Verlag, Berlin, pp. 167–81.
- Wettschereck, D., Aha, D. (1995) Weighting features, in: *Proceedings of the 1st International Conference on Case Based Reasoning (ICCBR 1995)*. Springer-Verlag.
- Wilke, W., Bergmann, R. (1998) Techniques and knowledge used for adaptation during case-based problem solving, in: *Proceedings of IEA-98-AIE, Lecture Notes in Computer Science*. Springer-Verlag, Berlin, Heidelberg, New York.
- Wilke, W., Smyth, B., Cunningham, P. (1998) Configuration techniques for adaptation, in: Lenz, M., Bartsch-Spörl, B., Burkhard, H.-D., Wess, S. (Eds.), *Case Based Reasoning Technology: from Foundations to Applications*, LNAI 1400. Springer-Verlag, pp. 17–50.
- Wilson, D. (2001) *Case-Base Maintenance: The Husbandry of Experience*. PhD thesis, Indiana University.
- Wold, E., Blum, T., Keislar, D., Wheaton, J. (1996) Content-based classification search and retrieval of audio. *IEEE Multimedia* 3(3), 27–36, Fall.

- Wold, E., Blum, T., Keislar, D., Wheaton, J. (1999) Classification, search, and retrieval of audio, in: Furht, B. (Ed.), Handbook of Multimedia Computing. CRC Press, chapter 10.
- Yao, Y.Y. (1995) Measuring retrieval effectiveness based on user preference of documents. Journal of the American Society for Information Science 46(2).
- Zhang, Z., Yang, Q. (1999) Dynamic refinement of feature-weights in CBR using quantitative introspective learning, in: Proceedings of the International Joint Conference in Artificial Intelligence, 1999.
- Zhang, H., Furht, B., Smoliar, S. (1995) Video and Image Processing in Multimedia Systems. Kluwer Academic Publishers, Boston.
- Zwicker, E., Zwicker, T. (1991) Audio engineering and psychoacoustics: matching signals to the final receiver, the human auditory system. Journal of Audio Engineering Society 39(3) 115–126, March.

Appendix A:

Services provided by Digital Service Providers

Table A-1 A Summary of services offered by Digital Service Providers

DSP	Affiliated to:	Record labels	No. of songs	Delivery model	Price	Format	DRM restrictions	Content Restrictions	Media player	Region	Services	Recommendation Service
iTunes	Apple computers	5 majors	200,000	Download	\$0.99 per song	ACC encoding at 128 mb, "a higher quality" format than mp3" – Apple spokesperson	"users have the same level of rights as people who bought CDs" – Apple. Unlimited CD burning; Transfer to an iPod mobile device. Encryption limits the number (3) of computers to which music can be transferred.	some copyright holders aren't letting Apple sell certain albums in their entirety – "partial albums"	iTunes Media Player, Mac OS X,	US	You can search before you pay unlike the subscription services. You can avail of a 30 second preview of each track.	"Users who bought this track also bought these tracks..." "Smart Playlist": a playlist template based on genre. You can avail of a 30 second preview of each track. these are not distributed to other users
Rhapsody (listen.com)	Listen.com acquired by Real Networks in April 2003	5 majors and over 50 independents	Over 300,000 songs = 20,000 albums (9,000 artists)	Streaming with facility to burn selected tracks to CD. No portable downloads.	\$9.99 per month subscription + \$ 0.99 per track to burn to CD	Windows Media 8. After Real Networks acquisition it is moving to a Real Networks Solution.	Streamed using windows Media DRM. Files cannot be downloaded, but can be burned to CD. Not able to transfer to mobile devices yet. After Real Networks acquisition it is moving to a Real Networks Solution.	Due to licensing restrictions not all tracks are burnable.	Rhapsody player which currently uses Windows Media components.	US	Create, name and save playlists to stream. You can send playlists to a friend. You can view most popular artists. Roaming profile.	Can stream a set of songs similar to a particular artist. Each week their editors create a new playlist suited to your taste in music.
PressPlay	Sony Music Entertainment and Universal Music Group. (Recently Acquired by Roxio Software)	5 majors plus independents	Over 300,000 songs	Unlimited streams/downloads. Portable downloads cost extra. See DRM restrictions.	\$9.95 per month subscription. + "portable" download charges. 5-pack = \$5.95 10-pack = \$9.95 20-pack = \$18.95	Secure Windows Media Player 9	Downloads are enabled only as long as the subscription is maintained. "Portable" downloads can be kept after subscription expiration. Transferable to portable devices that support Net MD1™ and secure WMA format. Can transfer media to one other computer.	Due to licensing not all tracks are burnable	Windows Media Player 9	US only	You can see what other members are streaming. Recommendation service allied to the search engine. (apparently ACF based). 'Build your own station' facility.	

Appendix B:

Services provided by Music Recommender Companies

Table A-2: Services provided by music recommender companies

Technology Provider	Content/Contentless	Knowledge Extraction	Recommendation techniques	Description of service	Clients
SavageBeast.com	content	Each song is analysed along 400 musical attributes – rhythm, lyrics, instrumentation and compositional qualities. Each song is worked on by at least one of their music analysts. <i>Does not</i> use machine-listening or other forms of automated data extraction	<i>Query by Instance</i> type Retrieval	Content based recommendation service - Powered by their proprietary Music Genome Project – “the most sophisticated taxonomy of music ever collected” Music library analysis service.	AOL Music Barnes and Noble Tower Records
MusicGenome.com	content	“Ground-breaking research in artificial intelligence and Music cognition”. After analysis each song is assigned a music DNA marker. It is not clear whether they use automated or human feature extraction techniques	A personal musical profile is built by analyzing the user’s response to a small sample of songs, or by monitoring his/her listening patterns. With each interaction the user’s profile is enhanced and “adapted,” thus tailoring more accurate music recommendations.	Content based recommendation service	Not available
MediaUnbound	content/contentless	Music experts mark up build up maps that correlate how close to each other various bands, sounds and songs are. Judging by the questions asked during bootstrap, these are features that have a cultural context.	Users engage in a personalization session where they rate songs as well as answering questions like ‘how do you like fm radio’	Song Characterization System Personalization Session - gathers preference information from users Selection Engine - selects lists of songs for each user Arrangement Engine - arranges songs into playlists using playlist heuristics	Pressplay
Agent Artists	content/contentless	Not specified – appears to be metadata augmented	‘Query by instance’ type retrieval User preferences are clustered into ‘styles’ which are described extensionally. Builds map of content relationships (by demographics) based on user preferences.	Automated preference capture Profile builder based on content meta-data Playlist management system Relational metadata creation	EMusic.com MP3.com Blockbuster VUNet USA Telstra.com Gracenote Inc. Addicted To Noise WindowsMedia.com Avant Go