Contextualize Your Listening: The Playlist as Recommendation Engine

Benjamin Fields

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
of the
University of London.

Department of Computing
Goldsmiths, University of London

2011

I certify that this thesis, and the research to which it refers, are the product of my own work, and that any ideas or quotations from the work of other people, published or otherwise, are fully acknowledged in accordance with the standard referencing practices of the discipline. I acknowledge the helpful guidance and support of my supervisors, Dr. Christophe Rhodes, Prof. Mark d'Inverno and Prof. Michael Casey.

Abstract

It is not hyperbole to note that a revolution has occurred in the way that we as a society distribute data and information. This revolution has come about through the confluence of Web-related technologies and the approaching-universal adoption of internet connectivity. Add to this mix the normalised use of lossy compression in digital music and the increase in digital music download and streaming services; the result is an environment where nearly anyone can listen to nearly any piece of music nearly anywhere. This is in many respects the pinnacle in music access and availability. Yet, a listener is now faced with a dilemma of choice. Without being familiar with the ever-expanding millions of songs available, how does a listener know what to listen to? If a near-complete collection of recorded music is available what does one listen to next? While the world of music distribution underwent a revolution, the ubiquitous access and availability it created brought new problems in recommendation and discovery.

In this thesis, a solution to these problems of recommendation and discovery is presented. We begin with an introduction to the core concepts around the playlist (i.e. sequential ordering of musical works). Next, we examine the history of the playlist as a recommendation technique, starting from before the invention of audio recording and moving through to modern automatic methods. This leads to an awareness that the creation of suitable playlists requires a high degree of knowledge of the relation between songs in a collection (e.g. song similarity). To better inform our base of knowledge of the relationships between songs we explore the use of social network analysis in combination with content-based music information retrieval. In an effort to show the promise of this more complex relational space, a fully automatic interactive radio system is proposed, using audio-content and social network data as a backbone. The implementation of the system is detailed. The creation of this system presents another problem in the area of evaluation. To that end, a novel distance metric between playlists is specified and tested. We then conclude with a discussion of what has been shown and what future work remains.

Acknowledgements

I would like to thank my supervisors, Christophe Rhodes, Mark d'Inverno, and Michael Casey. Each of you have provided an invaluable piece of the support structure that enabled the work described in this thesis (and the writing itself). Thanks to everyone who assisted in the editing of this document, especially Dan Stowell for early feedback, David Lewis for copyediting, and for the helpful critique from my examiners Anssi Klapuri and Stefan Rüger. While this thesis is much improved with their review, any remaining errors are my own.

Thanks to Tim Crawford for giving my technical work musical context. Thank you to everyone in the Intelligent Sound and Music Systems at Goldsmiths University of London. In no particular order: Mike, Alex, Geraint, Jamie, Bruno, Mick, Richard, Daniel, Ray, Daniel, Polina, Daniel, Hamish, Ollie, Marcus, and Alastair. Thanks to Kurt Jacobson for all your efforts on Myspace and communities. Thank you to Yves Raimond for introducing me to Linked Data and Hackdays. Thanks as well to the many people at the Centre for Digital Music, Queen Mary, University of London that I have had the great pleasure of working with these last few years, through multi-institution grants or other contexts: Matthew, Matthias, Amélie, Andrew, Chris, Mark, Chris, Mark, Adam, Mark, Katy, and Sefki. Thanks to Claudio Baccigalupo for many thoughts on structured data from the Web. Thank you to Paul Lamere for your tireless work in all things music informatics, especially your collaboration on our tutorial. This work would not have been possible without the support of: the Engineering and Physical Sciences Research Council through the Online Music Recognition and Search 2 grant (EP/E02274X/1); the Goldsmiths Department of Computing overseas research fund; a grant from the Audio Engineer Society Educational Foundation; and the Andrew W. Mellon Foundation supported Networked Environment for Music Analysis project.

Finally thanks to my family. To my parents, for imparting the intellectual curiosity required and unconditional love and support (and editing!). To Becky, for tolerating my odd work hours, and listening to my ideas. Consider the favour returned.

Contents

List of Figures

List of Tables

Publications

B. Fields and M. Casey. Using audio classifiers as a mechanism for content based song similarity. In *International Convention of Audio Engineering Society*, New York, NY, USA, October 2007.

M. Mauch, S. Dixon, C. Harte, M. Casey, and **B. Fields**. Discovering chord idioms through Beatles and real book songs. In *International Confernce on Music Information Retrieval (ISMIR)*, Vienna, Austria, September 2007.

K. Jacobson, **B. Fields**, M. Sandler and M. Casey. The effects of lossy audio encoding on genre classification tasks In *International Convention of Audio Engineering Society*, Amsterdam, Netherlands, May, 2008.

B. Fields, K. Jacobson, M. Casey, and M. Sandler. Do you sound like your friends? exploring artist similarity via artist social network relationships and audio signal processing. In *International Computer Music Conference (ICMC)*, Belfast, United Kingdom, August 2008.

B. Fields, K. Jacobson, C. Rhodes, and M. Casey. Social playlists and bottleneck measurements : Exploiting musician social graphs using content-based dissimilarity and pairwise maximum flow values. In *International Conference on Music Information Retrieval (ISMIR)*, Philadelphia, Pennsylvania, USA, September 2008.

K. Jacobson, **B. Fields**, and M. Sandler. Using audio analysis and network structure to identify communities in on-line social networks of artists. In *International Conference on Music Information Retrieval (ISMIR)*, Philadelphia, Pennsylvania, USA, September 2008.

B. Fields, C. Rhodes, M. d'Inverno. Using song social tags and topic models to describe and compare playlists. In *Workshop on Music Recommendation and Discovery (WOMRAD)*, Co-Located with *ACM Recommender Systems (RecSys)*, Barcelona, Spain, September 2010.

K. Page, **B. Fields**, B. Nagel, G. O'Neill, D. De Roure and T. Crawford. Semantics for music analysis through linked data: How country is my country? In *IEEE International Conference on eScience*, Brisbane, Australia, December 2010.

B. Fields, K. Jacobson, C. Rhodes, M. Sandler, M. d'Inverno and M. Casey. Analysis and Exploitation of Musician Social Networks for Recommendation and Discovery. (*accepted*) IEEE Transactions on Multimedia.

B. Fields, C. Rhodes, M. d'Inverno. Automatic Group-Interactive Radio using Social-Networks of Musicians. In *International Conference on Weblogs and Social Media (ICWSM)*, Barcelona, Spain, July 2011.

Demonstrations and Talks

(*invited talk*) **B. Fields** and K. Page. The Semantic Web and Why You Should Care. Given at *Integrating Digital Library Content with Computational Tools and Services Workshop*, Co-Located with *ACM and IEEE Joint Conference on Digital Libraries (JCDL)*, Austin, Texas, USA, 19 June 2009.

(*installation*) R. Stewart, M. Magas, and **B. Fields**. decibel 151: Collaborative Spatial Audio Interactive Environment. Presented at *ACM SIGGRAPH Art Gallery*, New Orleans, Louisiana, August 2009.

(*tutorial*) C. Baccigalupo and **B. Fields**. Mining the social web for music-related data. Given at *International Conference on Music Information Retrieval (ISMIR)*, Kobe, Japan, 26 October 2009.

(*demonstration*) **B. Fields** and C. Rhodes. An Audience Steerable Automatic Music Director for Online Radio Broadcast. Presented at *International Conference on Music Information Retrieval (ISMIR)*, Kobe, Japan, October 2009.

(*presentation*) **B. Fields**, K. Jacobson, C. Rhodes, M. Sandler and M. Casey. Analysis and Exploitation of Musician Social Networks for Recommendation and Discovery. Given at *IEEE THEMES,* co-located with *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Dallas, Texas, March 2010.

(*tutorial*) **B. Fields** and P. Lamere. Finding a Path Through the Juke Box: The Playlist Tutorial. Given at *International Conference on Music Information Retrieval (ISMIR)*, Utrecht, The Netherlands, 9 August 2010.

(*demonstration*) K. Page, **B. Fields**, B. Nagel, G. O'Neill, D. De Roure and T. Crawford. Semantics for Signal and Result Collections through Linked Data: How Country is my Country? Presented at *International Conference on Music Information Retrieval (ISMIR)*, Utrecht, The Netherlands, August 2010.

(*demonstration*) K. Page, **B. Fields**, B. Nagel, G. O'Neill, D. De Roure and T. Crawford. Semantics for music researchers: How country is my country?. Presented at *International Conference on the Semantic Web*, Shanghai, China, November 2010.

Chapter 1

Introduction

> "Over the piano was printed a notice: Please do not shoot the pianist. He is doing his best."
> –Oscar Wilde, *Personal Impressions of America*, 1883

The way we as a society listen to and consume music has changed radically in the last century and is still very much in flux. 150 years ago, if you wanted to listen to a piece of music, it needed to be performed where you could physically hear it. This performance could be formal, informal or even mechanised (*e.g.* the player piano).

Recorded music then completely changed how we listen to music. Recordings separate the listener from the performer; this separation creates many new ways of consuming music. For much of the 20^{th} century if you liked a piece of music, you would go to the store and buy a piece of physical media (a record, tape or CD) containing the recording of that piece of music, much as you would buy a frying pan or a carton of milk.

The Web is now completely transforming how we interact with music again. This is principally happening through a radical change in the distribution of recorded music. The Internet and the Web on top of it have made the distribution of any sort of information orders of magnitude cheaper than was previously possible. This reduction of the cost of distribution in recorded music is contributing heavily to the steady decline in the sales of music [Blow, 2009]. A growing body of data shows[1] that people are replacing their music buying (which has shifted to downloads of digital music) to music streaming, on demand. This is especially true among the young. This represents at least as big a shift in the way people think about music as occurred with the onset and widespread commercialisation of recorded music. The digital audio available for

[1] For example, market data about the US: http://evolver.fm/2010/11/11/americans-now-stream-as-much-music-as-they-download/ or France: http://www.mediametrie.fr/comportements/communiques/la-musique-sur-internet-a-trouve-son-public-aupres-des-jeunes.php?id=353

immediate download in the "record store", e.g. Apple's iTunes or Amazon.com, represents more variety and coverage than the largest physical records store that predate them and these inventories are available anywhere where there is an internet connection, rather than just population centers. Newer streaming-on-demand services such as Spotify[2] or Play.me[3] push this further, completely removing the idea of purchasing music and making their large catalogue one and the same as a subscriber's personal collection of music.

It is this rapidly-changing music environment that forms the backdrop of this thesis. Here we find both new problems and opportunities. When you can listen to any of millions of songs at any moment, how can you find the right music for the moment? How can you most effectively discover new music you will enjoy? These questions used to be constrained; given the music available to me, what should I listen to or what is new that I would like? The new reality of on-demand streaming and, to a lesser extent, digital downloads has removed this constraint. However, in its place we have a web of information, about the artists producing this music and their fans.

Taken together this requires better tools for music recommendation and discovery; it also makes new data sources available to enable the development of these tools. It is in this context that we present this thesis. The objective of this thesis is to make a clear contribution to the complimentary disciplines of information retrieval and recommender systems in the domain of music.

1.1 Definitions

This thesis covers work which is interdisciplinary, and, while narrow in focus, the scope of relevant background material is wide. In order to discuss ideas from multiple disciplines and fields at once and do so coherently, it is necessary to define a few terms.

A **recommender system** is a technique or method that presents a user with suggested objects for consumption based on past behaviour. For the purpose of this thesis we take a recommender system to encompass the suggestion of an object for any applicable domain-specific interaction with that object. In the case of music, we would describe a system that suggests songs for a user to listen to, based on data about the listener, the songs or both, as a music recommender system.

Information retrieval is the interdisciplinary field of searching for documents that are deemed relevant from among a collection. The collection can take many forms, including a relational database, a digital library repository

[2] http://spotify.com
[3] http://play.me

or the web. When considering *music* information retrieval, the *document* is typically a piece of music (or a segment of a piece) from a collection, where relevance is determined based on the particular use case of a system (*e.g.* when searching for remixes by example, a system uses a statistical analysis of digital audio signal looking for a high threshold of similar content).

In lay terms, *playlist* is incredibly overloaded, covering everything from an unordered set of songs (the classic definition in radio) to a file on disk serialising a specific ordered list of songs stored locally (for example, serialised XSPF[4]) to everything in between. In this thesis, *playlist* is taken to encompass all of these meanings (and a bit more).

We explicitly define a **playlist** as any group of songs, typically ordered, that are meant to be listened to or otherwise consumed together. This definition covers some things that are not commonly considered as playlists. This includes live performances of groups of songs such as the set of songs a band plays at a particular concert or a programme performed by a symphony orchestra. It also includes cases from recorded works such as a commercially released album. This may cover both conceptual models of a group of songs or instance and serialisations of groups of songs.

Throughout this work **fitness** is used to describe the suitability of a playlist to a given situation (as defined through explicit user input (e.g. seed song) plus context such as environment or user profile). That is, fitness is a measure of how well a given playlist matches (or *fits*) some request, the combination of explicit user input and implicit context.

1.2 Aims

The work described in this thesis is driven by three specific aims:

Extend automatic similarity methods to encode information from the web

The amount of openly accessible data related to music is vast in both size and reach. Though ostensibly describing music itself, these data principally describe the producers and consumers of music, artists and fans. This can be described as a relational web and its members compared using techniques from the mature field of graph theory. In parallel to this, there are many ways to find the similarity of the described content of music: via textual metadata, comparisons are made of artist name, genre label, title of work, and album name among other descriptors; via a statistical description of the content of audio signal representation of a musical work such as timbre or pitch. In this

[4] http://xspf.org/

work we aim to integrate these two approaches, social and content, to create a hybrid notion of similarity.

Design a music recommendation system using time and social contexts, inspired by terrestrial music radio

With the improved encoding of similarity made available by meeting our first aim, we set out to build an end user music recommendation application that takes advantage of this information. In particular we take as our use case that of a request radio show. This use case allows for interaction from listeners to the system by way of periodic requests for specific songs.

Improve and extend objective measures for playlists

Current literature concerning playlist generation fails to describe robust objective measures for playlist fitness. In order to improve this, we aim to specify novel means to compare playlists. In particular, our measure is derived from existing data through processes that require minimal human intervention. In this way, we can come closer to human user evaluation results while minimising the costly use of human test subjects.

1.3 Focus

In order to form a tractable set of problems, in addition to our aims we detail the focus of this work via the following constraints.

While there are many ways to describe music artists and the songs they produce, for this work we are concerned with a subset. We constrain this work to using self-generated textual descriptors (*e.g.* "hip-hop" or "guitar driven" as self-labelled) and relational links (*e.g.* "friends" or "followers") to describe artists. Other artists descriptors (*e.g.* expert generated labels) will not be directly used, though this work could be extended to consider many of them. When considering songs this work will limit to two forms of description: digital audio content-derived features and social tags. Again, there are other descriptors which may be useful (*e.g.* symbolic representation) though they are out of the scope of this work. Taken together these restrictions allow a dataset to be drawn from one of a number of self-publishing popular music websites (*e.g.* Myspace or Soundcloud) without a need for further resources

Additionally, though our similarity work is applicable to many different playlist generation techniques in many different environments, we focus on generating playlists between a specified start and end song for radio applications. Further, we model the song-to-song relationships for the purpose of playlist generation as a fully connected network rather than, e.g. a multidimensional metric-space. This is done to prioritise deep rather than wide applicability of

the similarity work to playlist generation and an end-user application.

1.4 Thesis Outline

To allow the reader a better understanding of the structure of this document, the following is an outline of this thesis by summarising the content of the remaining chapters.

Chapter 2: Playlists and Program Direction. We survey the state of the art in playlist tools and playlist generation. A framework for types of playlists is presented. We then give a brief history of playlist creation. This is followed by a discussion of music similarity, the current state of the art and how playlist generation depends on music similarity. The remainder of the chapter covers a representative survey of all things playlist. This includes commercially available tools to make and manage playlists, research into playlist generation and analysis of playlists from a selection of available playlist generators. Having reviewed existing tools and generation methods, we aim to demonstrate that a better understanding of song-to-song relationships than currently exists is a necessary underpinning for a robust playlist generation system, and this motivates much of the work in this thesis.

Chapter 3: Multimodal Social Network Analysis. We present an extensive analysis of a sample of a social network of musicians. First we analyse the network sample using standard complex network techniques to verify that it has similar properties to other web-derived complex networks. We then compute content-based pairwise dissimilarity values using the musical data associated with the network sample, and the relationship between those content-based distances and distances from network theory are explored. Following this exploration, hybrid graphs and distance measures are constructed and used to examine the community structure of the artist network. We close the chapter by presenting the results of these investigations and consider the recommendation and discovery applications these hybrid measures improve.

Chapter 4: Steerable Optimizing Self-Organized Radio. Using request radio shows as a base interactive model, we present the Steerable Optimizing Self-Organized Radio system as a prototypical music recommender system along side robust automatic playlist generation. This work builds directly on the hybrid models of similarity described in Chapter 3 through the creation of a web-based radio system that interacts with current listeners through the selection of periodic requests songs from a pool of

nominees. We describe the interactive model behind the request system. The system itself is then described in detail. We detail the evaluation process, though note that the inability to rigorously compare playlists creates some difficulty for a complete study.

Chapter 5: A Method to Describe and Compare Playlists. In this chapter we survey current means of evaluating playlists. We present a means of comparing playlists in a reduced dimensional space through the use of aggregated tag clouds and topic models. To evaluate the fitness of this measure, we perform prototypical retrieval tasks on playlists taken from radio station logs gathered from Radio Paradise and Yes.com, using tags from Last.fm with the result showing better than random performance when using the query playlist's station as ground truth, while failing to do so when using time of day as ground truth. We then discuss possible applications for this measurement technique as well as ways it might be improved.

Chapter 6: Conclusions. We discuss the findings of this thesis in their totality. After summarizing the conclusions we discuss possible future work and directions implied by these findings.

Chapter 2

Playlists and Program Direction

> "Now, the making of a good compilation tape is a very subtle art. Many do's and don'ts. First of all you're using someone else's poetry to express how you feel. This is a delicate thing."
> – Nick Hornby, *High Fidelity*, 1995

In this chapter we discuss the playlist, in its many forms and uses, and lay out the case for the playlist as an ideal construct for music recommendation and discovery. We consider playlist generation as a delivery-oriented form of a recommender system. It becomes apparent that the construction of playlists are intricately tied to an understanding of how member songs relate to each other when examining the history of playlist generation. Therefore, we include a discussion of the most well-understood of song-to-song relationships: music similarity. We discuss notions of similarity that are automatically generated as this view of similarity currently informs playlist generation techniques. Further, we discuss what is meant when a playlist is described as 'good' or 'bad'. We then survey existing techniques, both commercially driven and academic, in the creation of playlists using music similarity and the bounds of subjective judgement as a foundation. We show that the currently narrow view of automatic similarity restricts the scope and depths of automatic playlist generation.

The use of context in the form of a playlist as a music recommender is the driving backdrop behind this research. Section 2.1 begins with an exploration of the playlist as recommender followed by a framework for considering various forms of real-world playlist use cases; a history of the development of the modern playlist is found in 2.2. This exposes a strong dependency on the awareness of pairwise song relationships, leading to a discussion of the underlying techniques in the field of automatic music similarity in Section 2.3. A brief discussion of how this is currently used to recommend music is presented in Section 2.4 and aspects of playlist measurement are considered in Section 2.5. This is followed by an explanation by way of representative examples of com-

mon methods for modelling and serialising playlists in Section 2.6; this sections also covers how legal structures affect playlists, especially those delivered via streaming audio over the internet. Using the preceding core of understanding, a survey is presented of deployed tools in Section 2.7 followed by a survey of research systems in Section 2.8. Finally, Section 2.9 discusses what this past research has solved well and where novel approaches are clearly necessary.

2.1 The Playlist as Recommender

Putting together a playlist is solving a particular music recommendation problem. This is due to the fact that the context of selecting and ordering a set of songs from a much larger collection is what separates playlist generation from other music information-retrieval tasks such as query by humming or automatic transcription. This allows a much wider set of existing literature to be exploited. While the idea of viewing a recommendation in context is not yet commonplace in the domain of music, it has been effectively used in generic recommender systems [Anand and Mobasher, 2007] and in text information retrieval [Dunlop and Crossan, 2000]. Most commonly, some predictive text systems use the entire preceding sentence to make more informed guesses about the word being typed [Cohen and Singer, 1999; Stocky et al., 2004], allowing more efficient use of larger dictionaries. In many respects the playlist can be viewed as an analog to a sentence of text. By observing and exploiting what has come previously (or what is known to come next) a recommendation system can eliminate many otherwise possible candidates for recommendation. Another relevant technique from context based recommendation is *boosting*. A system using boosting takes a recommendation that is acceptable independent of any context and "boosts" it to a much stronger recommendation with presented in context with supporting recommended objects [Hayes and Cunningham, 2004]. In the music domain, this is consciously performed by club DJs and radio show presenters [Brewster and Broughton, 2006]. It may also be present in other forms of human playlist generation, though this is less documented.

2.1.1 Categorising Playlists by Producer and Consumer

A playlist defines a wide variety of forms of music presentation. It can encompass nearly any ordered list of songs. To discuss how playlists function in different contexts and situations, it is beneficial to categorise playlists based on common traits. What follows divides various real-world forms of playlists sorted by the relationship between the entity creating the playlist (the playlist's *producer*) and the intended entity to listen to the playlist (the playlist's *consumer*). Examples of each category are discussed along with common characteristics of each class. These categories and their common members are shown in Table 2.1.

Expert to Listener	Peer to Peer	Listener to Self
Club DJ	Mixtape	Digital Library Playlists
Radio DJ	Web Published Playlists	Portable Player Playlists
Commercial mix CD		

Table 2.1: Various kinds of playlists classified based on the relationship between producer and consumer.

2.1.1.1 Expert to Listener: The Curator

This category encompasses any and all cases where the playlist is generated by a professional with the intended playback to occur via some medium which will allow many people to listen concurrently. Common examples that fit in this category include various forms of performance by disc jockey (DJ), (*e.g.* Radio Specialty Show, Standard Rotation Music Radio Show, Club DJ). In each of these use cases the basic idea is the same. From some finite set of songs a subset is selected and ordered based on the particular requirements and goals of the use case. The key factors that tend to subdivide this category are the total size of the collection from which songs are drawn and particular kind of feedback experts receive from their audience.

There is another form within this category that has emerged with the widespread adoption of personal digital music player (*e.g.* iPods), the *book of playlists* [Ellingham, 2007; Lynskey, 2008]. These are playlists compiled by experts with content similar to a greatest hits CD or a compilation CD based on a topical idea. Unlike a compilation album, the experts (*e.g.* authors) do not provide the tracks in these playlists; to listen to these playlists the reader must find the tracks themselves.

2.1.1.2 Peer to Peer: A Contextualised Recommendation Among Friends

The Peer to Peer playlist is inherently social. These use cases encompass devices for peers to recommend not simply music but a context into which the recommended music is best understood or appreciated. This is encapsulated in the *mixtape*, a cassette tape of tracks recorded via playback of various records or other sources (*e.g.* radio) creating a personalised compilation. In the internet age, the same idea is realized over the internet, where a number of websites and internet-aware applications allow for the broadcasting and sharing of playlists (*e.g.* `http://mystrands.com`, `http://webjay.com`, iTunes).

2.1.1.3 Listener to Self: Everyone is a DJ

The simplest relationship between producer and consumer of playlists is when they are the same entity. This is seen most readily with a user of digital

media management software, such as iTunes or Windows Media Player, creating playlists within the software. The listener to self use case is less about discovery of new media and more concerned with exploiting known content to its fullest, since all songs are from the personal collection of the listener.

2.1.1.4 Functional playlists

Functional playlists encapsulate selection and order of music for playback in non-listening environments. In other words, a functional playlist is any playlist that serves as *background* while some other function is taking place. This can best be seen commercially in the work of Muzak[1]. While this is perhaps the most common way playlists are used in the modern world [Lanza, 2004], it is outside the scope of this document. Functional playlists are the furthest a *playlist* gets from a *music recommender system* as the goal is for the listener to do whatever other task they were doing anyway, with the music only subconsciously noticed [Lanza, 2004]. While there are certainly recommender system strategies that could automate the construction of this sort of playlist, the use case relies on significant environmental context in place of user taste and history, separating it completely from the other forms discussed in this section.

2.2 A History of Playlist Generation

Before discussing existing techniques in automatic playlist generation, it is useful to consider the history of manual or human generated playlist generation. For this we divide recent history based on trends in the construction of playlists and their driving forces. It should be apparent that these divisions are aligned with various technological innovations which lead to cultural changes in people's conceptions of music and narrative.

2.2.1 Before Recorded Music

In 1850's London a transition was taking place in the way concert programs were put together. Prior to this time a concert program was assembled in order to maximise coverage of taste. The pieces and instrumentation would vary wildly throughout the concert, with the cultural expectation that the audience would listen to a subset of interest, attending as much for the social experience as music itself. Typically these concerts were quite long, with four to six hour or longer runtimes not uncommon.

It was during the 1850s that concert programs as they are seen in modern orchestral and symphonic settings began to take form [Weber, 2001]. Concerts became more focused; pieces were selected to meet an idea or notion that program director wished to express with the pieces and their ordering. Here we

[1] http://www.muzak.com

begin to see the idea of a concert's program being a *curated* set of musically works rather than simply selected to maximise attendance (not that commercial aspects were totally absent). Further innovation both before and during the time of radio would come to define many of the norms and conventions of concert programming including the use and mixing of familiar and novel composers in coherent programs [Kremp, 2010]. It is here that we see the birth of the playlist, though it would be many decades before the term came into use.

2.2.2 Early Radio

The next leap forward toward modern playlist generation was driven by technological innovation occurring during the late 19th and early 20th century. This innovation was principally in two parallel areas: wireless communications (e.g. broadcast radio) and audio recording and playback.

The core transmission and reception technology behind radio was discovered and harnessed in the 1890s, leading to a series of patents, most notably [Edison, 1891] and [Marconi, 1897]. Just prior to these developments in radio the core mechanisms for practically recording audio on a medium for playback were also being perfected; first on cylinders [Edison, 1878] and then on acetate discs which would eventually become the dominant format.

The emergence of these two technologies, radio and phonographs, laid the groundwork for music to be broadcast to much larger audiences and without the physical presence of the performing artists, thus massively increasing what was possible in curation of ordered sets of music. The first broadcast of a musical program is commonly attributed to Canadian Reginald Fessenden [Brewster and Broughton, 2006, p. 2] who, in 1906 broadcast a program featuring a mix of live musical performance (himself on violin), the playback of a phonograph record and a reading from the Gospel of Luke. Over the next few years, there were a few other one-time experimental attempts at musical broadcasting; however the first continuous broadcast of musical material over radio was started by Frank Conrad in 1920 [Levinthal, 1998]. These early broadcasts laid the groundwork for the mass-media playlist broadcasting that would follow.

2.2.3 Post-War Radio

As radio began to settle as a medium certain genres were pushed and emerged, especially rock and roll and R'n'B [Wall, 2007]. It was during this time that "playlist" was first used to describe (unordered) sets of songs [Brewster and Broughton, 2006, p. 20]. Much of the programming became personality-driven with considerable variation in how music-focussed the personalities were. One end of the spectrum was represented by hosts such as John Peel, who in many ways personified the notion of curating a set of songs. The other end of this

spectrum of radio hosts were the hosts of "Top 40" and countdown shows the best known of which is perhaps Casey Kasem. The programming on these shows was almost exclusively dictated by external constraints (i.e. sales) and as such the hosts were employed almost exclusively as voice talent.

2.2.4 The Emergence of the Club DJ

In the mid 1970s disc jockeys (DJs) in nightclubs and discos began to use two turntables with a mixer rather than a single turntable to minimise transition time between records during playback. This led to the birth of the concept of *continuous mixing* or the the elimination of space between songs played back in sequence. DJs did not want dancers to notice song transitions leading to techniques such as *beat matching*, where the tempo of two songs is made equal, and *phrase alignment*, where the phrase boundaries in songs are synchronized in time, were pioneered.

This idea was pushed further with the formation of hip-hop culture, as DJs became live remixers, turning the turntable into an instrument [Asante, 2008]. At the same time, club DJs started to become the top billing over live musicians, the curator becoming more of a draw than the artist [Brewster and Broughton, 2006, pp. 104-120].

2.2.5 The Playlist Goes Personal

Club and hip-hop culture continued to grow, while the emergence of portable audio devices drove the popularity of cassette tapes. The usage of tapes became wide spread, in turn leading to reordering and combining of disparate music material into *mixtapes* [Bull, 2006]. Mixtapes themselves were traded and distributed socially, providing a means for recommendation and discovery. They were also drivers of social interaction and larger culture, including fictional works such as Hornby [1995].

In hip-hop, the term mixtape described the recordings of DJs, featuring novel mixes and leading to current phenomenon of the *mixtape*, *mixset* and *mix CD* (now most commonly on CD or other digital media). It was through the sale and distribution of these mixes that DJs could make a name for themselves, using this medium to demonstrate their abilities.

2.2.6 Now With Internet

The creation and massive increase in uptake of the Web along with psychoacoustic audio compression (most notably the MPEG-1 and MPEG-2, layer 3 (MP3) standards [ISO/IEC 11172-3, 1993; ISO/IEC 13818-3, 1998]) allowed for practical sharing of music over the Internet. This brought the culture of mixtapes for physical sharing to the non-place of the internet [Freire, 2008]. By

removing the requirement for physical exchange, specialist mixtapes as playlists are now sustainable.

In addition to the peer to peer exchange of music and playlists, streaming-over-internet radio has emerged with the support of the same technologies. These internet-based radio stations are somewhat similar in format to traditional terrestrial radio. However, there are two significant changes affecting programming of these streamed radio stations. The internet allows a station to "broadcast" to anywhere in the world, massively increasing the potential pool of listeners for a station. At the same time, the barrier for entry (*e.g.* the cost to start broadcasting a stream) is significantly reduced. Together these two factors have the result of creating more stations that broadcast more specialised content, though these effects are somewhat mitigated through copyright licensing and statutes which are discussed in Section 2.7.3.

Finally, there has been a recent trend toward common web-based storage (colloquially *on the cloud*) of playlists. Some of these services provide listenable content or links to buy content. A few popular examples include Spotify[2], Play.me[3] and Mog[4].

2.3 Music Similarity

Central to the emergent modern understanding of a playlist is a notion of structure, from a minimal internal coherence to a complex narrative. Understanding the structures of playlists for generation and construction depends on understanding the *music similarity*, or the way and amount by which music objects are similar, amongst the songs making up a giving playlist and those in the collection from which the playlist is drawn. It is therefore crucial to have a core of knowledge about music similarity.

Current understandings of the similarity between objects is heavily informed by Tversky [1977]. In this work, objects are considered as sets of features and an object's similarity with another is found by comparing the features which are the same (or in some case close) in value. For example, if there are two toy balls A and B, and ball A can be considered as the set of features $(small, red, rigid)$ and ball B can be considered as the set of features $(small, blue, rigid)$; balls A and B are then taken as similar via the features of size (both small) and hardness (both rigid), while they differ in their respective colours (A being red, while B is blue).

When people consider the similarity of music objects, applying the same

[2] http://www.spotify.com
[3] http://www.playme.com/
[4] http://mog.com/

approach is straightforward. Two pieces of music may be close in tempo and rhythmic structure, but have different instrumentation; this would make them similar by two features in a set of three. Music, being made up of both acoustic signal and the brain's interpretation of that signal, can have features drawn directly from the signal content or from other factors that impact musical object interpretation by the brain [Hargreaves and North, 1999; Krumhansl, 1995]. Many methods have been explored for content-based music analysis, attempting to characterise a digital audio signal of music by its timbre, harmony, rhythm, or structure. One of the most widely-used methods is the application of Mel-frequency cepstral coefficients (MFCC) to the modelling of timbre [Logan, 2000]. In combination with various statistical techniques, MFCCs have been successfully applied to music similarity and genre classification tasks [Berenzweig et al., 2004; Logan and Salomon, 2001; Pampalk, 2006]. Typically a digital audio file is split into *frames* with lengths on the order of $100ms$. A set of MFCCs can then be generated for each frame of audio in the following way [Logan, 2000]:

1. Apply a window to filter frame edge error. Then take the Fourier transform of the signal.

2. Transpose the magnitude powers of the resulting spectrum onto the mel scale. The mel scale, which approximates human listening precision, is defined as [O'Shaughnessy, 1987]

$$m = 2595 \log_{10} \left(\frac{f}{700} + 1 \right) \tag{2.1}$$

   where m is the result pitch in mels and f is the input frequency in Hertz. The mel scale results in pitches that are approximately linear with respect to frequency until 700 Hz and thereafter the relationship is approximately logarithmic. Typically this scaling is performed using triangular overlapping windows.

3. In each of the resulting mel frequency bins, take the log power.

4. Treat the list of mel scaled log powers as signal and take a discrete cosine transform (DCT). The DCT is defined as

$$X_k = \sum_{n=0}^{N-1} x_n \cos \left[\frac{\pi}{N} \left(n + \frac{1}{2} \right) k \right] \qquad k = 0, \cdots, N-1 \tag{2.2}$$

   where N is the number of bins used in the mel scale transposition of step 2.

5. The amplitudes of the resulting spectrum are MFCCs

In addition to MFCC, a number of other features are used in some automatic similarity algorithms, for simple measures such as zero-crossing counts to low-level frame-based features similar to MFCCs in complexity such as chromagrams. An exhaustive survey of these features is beyond the scope of our discussion. An excellent exhaustive treatment of the historical use of features for music similarity can be found in Pampalk [2006].

A common approach for computing timbre-based similarity between two songs or collections of songs creates a statistical model describing the MFCCs for an entire song and comparing these models using a statistical distance measure. The simplest of these models is a single mean and variance for each coefficient of the MFCC vector across a piece of digital audio. A more complex generalisation of this technique is a Gaussian Mixture Model (GMM) where multiple Gaussians distributions are weighted and combined to describe arbitrary distributions. A common way to calculate distance between the resulting statistical models is via the Kullback-Leibler divergence (KL divergence). For two discrete random variables P and Q it is defined as [Kullback and Leibler, 1951]:

$$D_{KL}(P||Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)} \tag{2.3}$$

The KL divergence is also frequently used in a symmetric form

$$D_{symKL}(P||Q) = D_{KL}(P||Q) + D_{KL}(Q||P) \tag{2.4}$$

though this simple symmetric form fails when either P or Q is zero and the other is not; more robust symmetric forms exist, most notably Jensen-Shannon Divergence [Lin, 1991].

Another measure of similarity is the Earth Mover's Distance (EMD), a technique first used in computer vision [Rubner et al., 2000]. It has been successfully used for music timbral similarity [Aucouturier and Pachet, 2004; Pampalk, 2006]. The EMD algorithm finds the minimum work required to transform one distribution into another: one distribution is taken as mounds of earth, while the other is taken as holes, work here refers to the movement of this earth into these holes. A number of other methods for finding the similarity between distributions are known and used for musical objects, most notably Markov Chain Monte Carlo (MCMC) [Andrieu et al., 2003].

In order to compare various automatic similarity methods, a task entitled

'Audio Music Similarity and Retrieval'[5] has been run for a number of years as part of The Music Information Retrieval Evaluation eXchange (MIREX) [Downie, 2006, 2008]. The remainder of this section will provide a formal description of the task, explanations of each participating algorithm, evaluation process and the 2010 results, as these methods represent the current state of the art in content-based similarity.

2.3.1 MIREX: Audio Music Similarity and Retrieval

Currently the Audio Music Similarity and Retrieval (AMS) task within MIREX has run four times: 2006, 2007, 2009 and 2010. While the task has not changed in any substantial way during that time, the task description and discussion of results that follows applies to the 2010 iteration of the task unless otherwise stated.

The remit for the task is quite simple: given a collection of 7,000 30-second clips of audio drawn equally from 10 genres, for each selected query track, find the 100 most similar tracks in the correct order and assigned the correct similarity score. No labels (e.g. metadata such as artist name or track title) are provided. Here *correct* refers to alignment with human listeners serving as evaluators, though a second evaluation using objective statistics is also employed. Both of these evaluation techniques will be explained in detail in Section 2.3.3.

2.3.2 Descriptions of Participating Algorithms

The 2010 AMS task compared eight algorithms from 14 contributors, including one that assigned a random score as a similarity, intended to serve as a control. These algorithms are named and their authors listed in Table 2.2.

2.3.2.1 MTG-AMS

Bogdanov et al. [2010] extract more than 60 descriptors from content including MFCCs, various song-wide and per-frame measures described by Bogdanov et al. [2009]. From this set of content descriptors, four distance measures are used: a Euclidean distance measure following the reduction of the dimensionality of the features using principal component analysis (PCA) as described in Cano et al. [2005]; KL divergence calculated on a Gaussian summary of frame-by-frame MFCCs similar to the algorithm detailed in the previous portion of this section; a tempo based distance, described in detail by Bogdanov et al. [2009]; and a *semantic classifier* based distance metric also described by Bogdanov et al. [2009]. These four distances were then joined using a linear combination with the coefficients selected a priori based on subjective user feedback, though at the time of writing the details of this user study and its

[5] past and current years' task descriptions, participant lists and results are available: http://www.music-ir.org/mirex/wiki/Audio_Music_Similarity_and_Retrieval

Algorithm name	Abbreviation	Contributors
MTG-AMS	BWL1	Dmitry Bogdanov, Joan Serrà, Nicolas Wack and Perfecto Herrera
PS09	PS1	Tim Pohle and Dominik Schnitzer
PSS10	PSS1	Tim Pohle, Klaus Seyerlehner and Dominik Schnitzer
RND	RZ1	None
cmbr_sim	SSPK2	Klaus Seyerlehner, Markus Schedl, Tim Pohle and Peter Knees
MarsyasSimilarity	TLN1	George Tzanetakis, Steven Ness and Mathieu Lagrange
Post-Processing 1 of Marsyas similarity results	TLN2	George Tzanetakis, Mathieu Lagrange and Steven Ness
Post-Processing 2 of Marsyas similarity results	TLN3	George Tzanetakis, Mathieu Lagrange and Steven Ness

Table 2.2: Participating algorithms and contributors in the 2010 run of the MIREX Audio Music Similarity and Retrieval task.

results have not yet been published. As can be seen in Table 2.3, this approach performed well, though not the best of the participating algorithms.

2.3.2.2 PS09

While Pohle and Schnitzer [2009] was run in 2010, it was submitted for the 2009 run of the task and having most closely aligned with human evaluation, was run again for comparative purposes. This algorithm uses two distance metrics, a timbral distance and a rhythmic distance, which are separately normalised then combined with equal weighting to form the overall distance metric. The timbral distance is generated through the use of MFCCs, Spectral Contrast Feature [Jiang et al., 2002] and estimates of the amount of harmonic and percussive content in a given frame of audio content [Ono et al., 2008]. These frame-by-frame values are then reduced into a single gaussian describing the distribution of features across the entire audio sample. The rhythmic distance is found with a variant of *fluctuation patterns* using a *cent/sone* representation [Pohle et al., 2009].

2.3.2.3 PSS10

The Pohle et al. [2010] system is mostly the same as Pohle and Schnitzer [2009]. It differs in the inclusion of an additional time-related component, similar to the rhythmic distance component used in the previous submission. In this

new component, periodicity is estimated using the full signal rather than *onset patterns* and *fluctuation patterns* as described by Pohle et al. [2009]. Further the weighting of the components has been adjusted to take into account the addition of a third. The weights in Pohle et al. [2010] are 70:20:10 for the timbral component, the new rhythmic component, and the old rhythmic component, respectively. Looking at Table 2.3, it can be see that Pohle and Schnitzer [2009] and Pohle et al. [2010] performed at a level that is a statistical a tie, in spite of the considerable increase in computational complexity in Pohle et al. [2010].

2.3.2.4 RND

This algorithm was included to be used as a random baseline in the task. It simply generated an $M \times M$ matrix of random values, where M is equal to the number of songs in the test set. Thus the "top 5" songs returned by a similarity query will be arbitrarily chosen.

2.3.2.5 cmbr_sim

To find similarity in Seyerlehner et al. [2010a], two distances are computed: block-level similarity and tag-affinity-based similarity. Block-level similarity is based on *block-level features*, or features derived from the concatenation of consecutive frames of log-scaled magnitude spectrum of audio. The length of these blocks varies depending on the feature. These features range from a simple sort of each band in a block to rhythmic patterns analysis measured by taking the FFT of a sufficiently large block (512 frames) to a block-wise correlation [Seyerlehner et al., 2010b]. To find similarity, each of the block-level features is compared separately using the Manhattan distance and joined using a technique called distance space normalization (DSN) as described by Pohle and Schnitzer [2007]. The difference applied here is to independently normalise each feature's distance matrix by subtracting the mean and dividing by the standard deviation. In the tag-affinity-based similarity, the songs are run through a number of pre-trained support vector machines as proposed in West et al. [2006]. These classifiers are trained to find tag affinity using *Magnatagatune* [Law and Ahn, 2009] and *CAL500* [Turnbull et al., 2008]. The affinity for each tag is then treated as a vector, one per song. The tag affinity based similarity is found by taking the manhattan distance between these tag affinity vectors. The overall distance between two songs is simply the addition of the two distances. This overall distance gave a metric which aligned more closely with the human evaluators than any other submitted algorithm, though its performance was within statistical significance of Pohle et al. [2010], Pohle and Schnitzer [2009] and Bogdanov et al. [2010].

2.3.2.6 MarsyasSimilarity, including post-processing 1 and 2

Tzanetakis [2010] describes the remaining three submissions: *TLN1*, *TLN2* and *TLN3*. *TLN1* is the base similarity algorithm with *TLN2* and *TLN3* post-processing the resulting distance metric. To compute the similarity Tzanetakis [2010] computes timbral, pitch and rhythmic features. To obtain timbral features, the system extracts MFCCs, spectral centroid, rolloff and flux for windowed audio frames [Tzanetakis and Cook, 2002]. A look-back mean and standard deviation of the last 40 frames is held while the raw frame data are discarded. This results in a 32-dimensional feature to describe the 30-second clip songs. Pitch information is found by calculating chromagrams for each frame. A running average for the entire song is used in the same manner as the timbral features. Finally the rhythmic component is found by calculating the *beat histogram*, a relative measure of the strength of tempo for all possible tempos between 40 and 200 beats per minute (BPM). These three component features are normalised from 0 to 1 then combined to form one vector per song. The distance matrix is formed by taking the Euclidean distance between every pair of vectors. Tzanetakis [2010] fails to explain the post-processing performed for *TLN2* and *TLN3*, however the results show little difference in evaluation, regardless of what was done.

2.3.3 Evaluation and 2010 Results

The evaluation process for the MIREX AMS task aims to test how well the participating algorithms align with human-measured music similarity over the same dataset. Each participating algorithm outputs an $N \times N$ matrix of numerical scores indicating how similar each song in the database of N songs is to all the other songs in the database. In the 2010 run of the task, 7,000 30-second song clips made up the dataset. Human listeners evaluate a subset of all possible combinations of songs. This subset was created via a random selection of 100 songs evenly distributed across 10 genres within the dataset; the human-evaluated dataset is made up of the five most similar results from each of these selected songs, across all participating algorithms. The human evaluators are then asked to quantify the similarity between these pairs of songs with two scores: a "broad" score of three phrases, *not similar*, *somewhat similar*, and *very similar*; and a "fine score", a numeric measure between 0 and 100 inclusive. Finally, the algorithms' performance is defined as how well the similarities produced by the algorithms align with those produced by the human evaluators. A bespoke web application called "The Evalutron 6000" is used for the human side of this evaluation [Gruzd et al., 2007].

The results for all the participating algorithms can be seen in Table 2.3

	Average Score		Genre Neighbourhood Clustering			
	Fine	**Broad**	**Top 5**	**Top 10**	**Top 20**	**Top 50**
BWL1	49.704	1.078	0.5319	0.5157	0.4987	0.4690
PS1	55.080	1.228	0.5900	0.5695	0.5474	0.5102
PSS1	54.984	1.212	**0.6186**	**0.6004**	**0.5786**	**0.5448**
RZ1	16.668	0.240	0.0831	0.0865	0.0867	0.0880
SSPK2	**56.642**	**1.248**	0.5910	0.5757	0.5583	0.5301
TLN1	45.842	0.940	0.4655	0.4478	0.4287	0.3995
TLN2	46.544	0.970	0.4797	0.4644	0.4448	0.4161
TLN3	46.604	0.968	0.4814	0.4664	0.4467	0.4179

Table 2.3: Aggregate results for each algorithm in the MIREX 2010 AMS task. Note that while SSPK2 had the best broad and fine scores, PSS1 did better in terms of genre neighbourhood clustering.

Algorithm	**Runtime**
BWL1	41h 40m
PS1	12h 50m
PSS1	6h 25m
SSPK2	19h 35m
TLN1	**1h 20m**
TLN2	1h 25m
TLN3	5h 00m

Table 2.4: The runtimes for each of the algorithms participating in the AMS MIREX task.

and Figure 2.1. These scores are the average of the song pairs retrieved by each algorithm. For the broad scores the three grading phrases are converted to the following numbers: *not similar*, 0; *somewhat similar*, 1; *very similar*, 2. In addition to the top line human evaluation, the similarity scores were also analyzed to see how homogeneous the top N results are based on a priori genre labels. While the best evaluation scores comes from Seyerlehner et al. [2010a], all the algorithms are very close in performance with the exception of the random baseline, which did considerably worse. As such it is worth considering the runtime performance of these algorithms given that accuracy was measured to be so similar. The runtimes of each algorithm are shown in Table 2.4. Here we see that the MarsyasSimilarity [Tzanetakis, 2010] is the fastest of the non-random algorithms, by a considerable margin.

While the AMS task and evaluation represents the broadest human evaluation of music similarity algorithms, it is not without flaws. Most notable is the

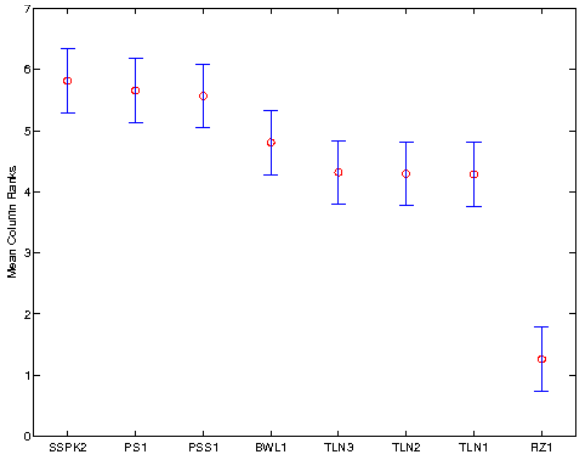


Figure 2.1: Aggregate fine scores of each participating algorithm in the MIREX 2010 AMS task.

use of the algorithms as a pre-filter in selecting song pairs that appear in the human evaluation. This leads to results that may miss similar pairs of songs in the whole dataset as all algorithms may have missed such items, in order to scope the evaluation to be achievable in a short time frame with a few dozen evaluators. On balance, though, these evaluations are a reasonable means to compare automatic music similarity.

2.4 Recommender Systems in Music

Whereas playlist generation is about listening and playback, recommender systems as classically defined are about consumption and acquisition. A *recommender system* is a mechanism for suggested new items for consumption based on past behavior. In general, this is achieved through optimization based on *user preference* of recommended objects for a given user. While a large number of methods have been explored to fulfill this generalized problem [Adomavicius and Tuzhilin, 2005; Resnick and Varian, 1997] the established base of recommender systems relies on a technique called *collaborative filtering*. In collaborative filtering users are compared based on how they have rated items in a collections (e.g. books, films, CDs). An unfamiliar item's rating can then be guessed based on the rating behavior of users with similar previous ratings on items that have been mutually rated [Herlocker et al., 2004]. Collaborative filtering methods can also be used with *passive user preference* inputs [Barkhuus and Dey, 2003]. These can include things like purchase history, webpage views or music play history.

Ratings are a common input method in many recommender systems. For

every item in a collection, users can *rate* the item by giving it a numeric score. Figure 2.2 shows the rating interface for items in Amazon.com's catalogue, which is typical of the ratings systems seen on many websites.

Figure 2.2: A screen capture of the Amazon.com page for the CD 'Tik Tok [Import Single]' by 'Ke$ha' showing the rating interface inside the overlaid red elipse. Here the stars are both the input (submit your rating by clicking on one of the five stars) and the output (the average rating for across all user rating for the item is show by the number of stars that are filled).

These ratings can then be used to drive a recommender system by determining similarity between users by using their rating behaviour. This similarity can then feed back to the initial user for predictive generation of ratings.

2.5 Finding a Good Playlist

When considering playlist generation it is important to understand the factors that can lead to a good playlist. Some of these factors relate to the songs within in a playlist. These factors can include a listener's preference for and familiarity with a song, song coherence, the variety of songs and artists in the set of songs and other less specific factors such as a song's *freshness* or *coolness*. Elements of the order in which the songs are arranged in a playlist also have an effect, including song transitions, the overall structure of a playlist and the occurrence of serendipity. In de Mooij [1997], an evaluation is conducted with 14 participants to ascertain which of eight factors has the greatest effect on a playlist. The results of this evaluation can be seen in Figure 2.3, showing that the actual song selection is rated by users as the most important factor. The

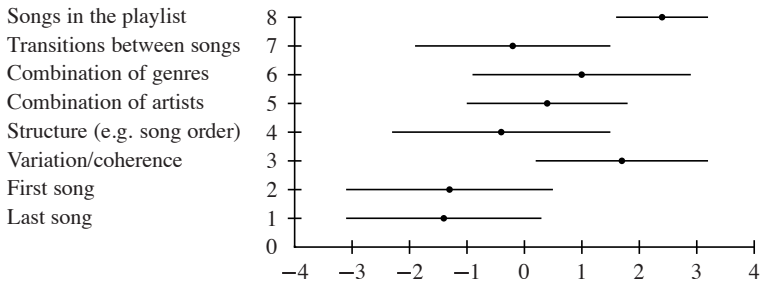


Figure 2.3: The relative importance of various factors in assembling a playlist, as found and published in de Mooij [1997].

work leaves open the question of what makes the songs right for the playlist. In particular, the idea of *musical taste* or the long-term, slow-to-change preferential commitment to a genre can have an impact. Musical taste can be inferred from recent listening history [Terveen et al., 2002; Voida et al., 2005]. Other elements can have a further effect on preference, especially a listener's mood and the context (i.e. just listening, driving, studying, exercise). One particularly important factor is how familiar a listener is with a song. It has been shown that people will often prefer listening to familiar songs that they like less than over non-familiar songs. Further familiarity is a significant predictor of choice even when controlling for the effect of liking, anticipated social perceptions and coolness [Ward et al., 2006].

2.5.1 Coherence and Order

Cunningham et al. [2006] exhaustively examined the motivating factors and organising principles driving the creating of playlists among an online community of hobbyists. Both the playlists and the surrounding conversations were analysed from the forum-driven website Art of the Mix[6] leading to set of common organising principles:

- Artist/Genre/Style – frequently a primer on the subject matter
- Song similarity – playlist by example
- An event or activity – to narrate a particular event such as a 'road trip'
- Romance – a mix to express one's feelings for another
- Message – e.g." 'quit being a douche,' 'cause I'm in love with you."
- Mood – a mix that moves from sad to uplifting

[6] http://www.artofthemix.org

- Challenge or puzzle – create a mix to satisfy artificial criteria
- Orchestration – a broad category to include instrumentation and other sound-related facets of performance
- Characteristic of the mix recipient – e.g. the types of songs required to complete mix for a given person
- Cultural references – a variety of popular and high cultural reference points

People make playlists for a variety of reasons as is clearly shown in this listing. Beyond selection, there are a number of methods that can be used to order the songs within the playlist set. These can include similarity score, other attributes derived from the acoustic (tempo, loudness, danceability), social attributes (popularity, hotness), mood attributes, theme or lyrics, alphabetical, chronological, random, optimising song transitions and novelty orderings. Some of these ordering technics are more effective in producing playlists that a listener might find interesting or enjoyable than others. An example of a less effective ordering driven by alphabetising is in Figure 2.4. One ordering technique that

Figure 2.4: A playlist within iTunes ordered alphabetically by title, courtesy of Paul Lamere.

is of interest is that of novelty ordering. In novelty ordering, arbitrary rule-sets are used to create both the song selection and the order of the songs. In practice this is very similar to the technique referred to in Cunningham et al. [2006] as "Challenge or Mood." Examples of novelty ordering that make use of

the Echonest's playlist Application Programming Interface (API)[7] have been discussed by Paul Lamere[8] including the following playlist, which meets two rules: the title of each song is shorter than the song before it by one character (shown in bold) and the last character of the title from song $n-1$ is the first character in the title for song n.

0. Tripping Down the Freeway - Weezer
1. **Y**er All I've Got Tonight - The Smashing Pumpkins
2. **T**he Most Beautiful Things - Jimmy Eat World
3. **S**omeday You Will Be Loved - Death Cab For Cutie
4. **D**on't Make Me Prove It - Veruca Salt
5. **T**he Sacred And Profane - Smashing Pumpkins, The
6. **E**verything Is Alright - Motion City Soundtrack
7. **T**he Ego's Last Stand - The Flaming Lips
8. **D**on't Believe A Word - Third Eye Blind
9. **D**on's Gone Columbia - Teenage Fanclub
10. **A**lone + Easy Target - Foo Fighters
11. **T**he Houses Of Roofs - Biffy Clyro
12. **S**anta Has a Mullet - Nerf Herder
13. **T**urtleneck Coverup - Ozma
14. **P**erfect Situation - Weezer

One area where song order is particularly important is in the playlists or sets constructed by dance DJs. Here the primary goal is to make people dance [Cliff, 1999]. This is done through selecting tracks that mix well, take the audience on journey and successfully integrate audience feedback [Silby, 2007]. These aims are furthered through the use of *seamless* song transitions.

The automatic system described by Cliff [2006] attempts to meet these needs, in part through the use of global tempo change and smooth song transitions. In various environments DJs select songs of varying tempo according to particular use cases. A selection of these can be seen in Figure 2.5.

[7] http://developer.echonest.com/docs/v4/playlist.html
[8] http://musicmachinery.com/2010/07/25/novelty-playlist-ordering/

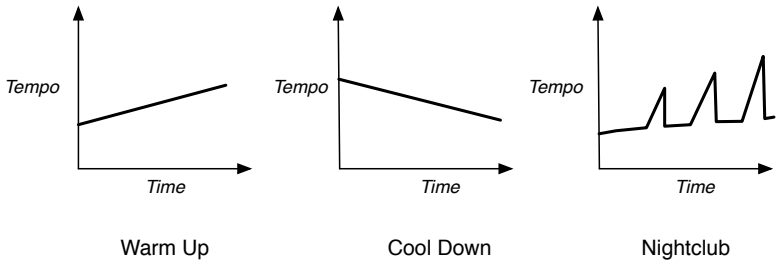


Figure 2.5: Three common examples of changing tempo curves across a Dance DJ set, labeled according to use [Cliff, 2006].

The system outlined in Cliff [1999, 2006] is designed to emulate these behaviours. Additionally the system creates smooth song transitions by using dynamic time warping to achieve beat synchronisation while mixing multiple songs.

2.5.2 The Serendipitous Lack of Order

In contrast to the notion that coherence comes from intentional or well-reasoned ordering, some work has shown that in certain circumstances a random order or *shuffle* of music for listening can result in a high degree of enjoyment [Leong et al., 2005]. In particular it has been shown that serendipity via a random order of one's personal collection can improve the listening experience. However if the random selection is drawn from a collection that is too large or too small the likelihood of a positive listening experience is reduced. The use of random ordering as a tool in listening was examined empirically in [Leong et al., 2006]. Through the use of their *Digital Music Listening Framework* the authors analysed 113 users' music libraries and listening habits looking at both organisational practices and ordering techniques. Of particular interest is the examination of preferred listening mode (shuffle or a sequential order) against the organisation of music content (constrained or unconstrained), shown in Table 2.5. The results from this work show a clear preference for random play when listening to personal collections, though again, the success of the random ordering in this case is clearly tied to the coherence and personal meaning of the underlying collection.

2.5.3 Summary

The creation of a good playlist necessitates the balance of many aspects. This can be considered as a tradeoff between poles across a number of axises. A simplified summary of these poles can be seen in Figure 2.6. The optimal balance between these and other related concepts is different for different listeners. Further, for a given user, listening habits and therefore optimal settings of these tradeoffs are affected by things like mood, time of day and environ-

		Content Organisation		
		constrained	unconstrained	
Preferred Listening	shuffle	22	69	91
	both	4	4	8
	sequential	13	1	14
		39	74	113

Table 2.5: Shuffle and sequential ordering as a preferred listening methods against both constrained and unconstrained content organisation methods [Leong et al., 2006].

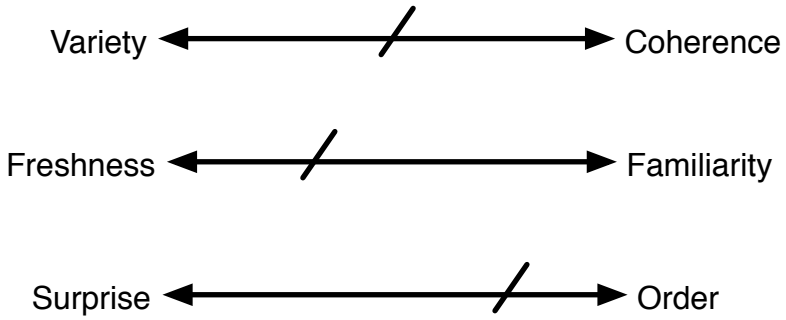


Figure 2.6: High-level concepts to balance in a playlist

mental context [Herrera et al., 2010]. Lastly, variety of some kind, whether from intentional ordering or random shuffle, is essential in the construction of a listenable playlist. This is self-evident from Figure 2.7.

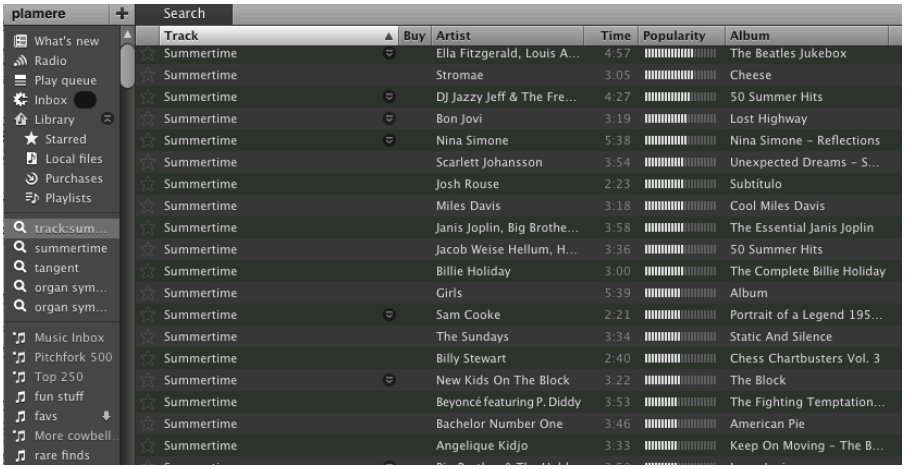
2.6 Formats and Legal Considerations

As playlists have proliferated with the adoption and spread of the internet, interchange and serialisation formats have been developed. Some notable examples of these formats[9] include M3U, which is simply a list of files (typically resolvable URLs), one per line, and XSPF (commonly pronounced 'spiff') an XML schema, an example of which is in Figure 2.8.

Additionally there is an emerging effort to formalize a data model around playlists and playlist construction using linked data Resource Description Framework (rdf) ontologies. The Playback Ontology[10] covers playlists as well

[9]Exhaustive and current lists can be found on various websites including http://microformats.org/wiki/audio-info-formats http://lizzy.sourceforge.net/docs/formats.html and http://gonze.com/playlists/playlist-format-survey.html among others.

[10]The ontology specification can be found here: http://smiy.sourceforge.net/pbo/spec/playbackontology.html and a brief walkthrough of initial use cases here: http://smiy.wordpress.com/2010/07/27/the-play-back-ontology/.



(a) A playlist of songs with the same title, from Spotify.

(b) Another same track playlist, this one with the seed song immediately repeated.

Figure 2.7: Two example playlists demonstrating the fallacy of assuming that similarity is equivalent to high fitness in playlist construction.

```
<?xml version="1.0" encoding="UTF-8"?>
<playlist version="1" xmlns="http://xspf.org/ns/0/">
  <trackList>
    <track>
      <location>http://example.com/song_1.mp3</location>
      <creator>Led Zeppelin</creator>
      <album>Houses of the Holy</album>
      <title>No Quarter</title>
      <annotation>I love this song</annotation>
      <duration>271066</duration>
      <image>
        http://images.amazon.com/images/P/B000002J0B.jpg
      </image>
      <info>http://example.com</info>
    </track>
    <track>
      <location>http://example.com/song_2.mp3</location>
      <creator>Emerson Lake & Palmer</creator>
      <album>Emerson Lake & Palmer</album>
      <title>Lucky Man</title>
      <annotation>This one too</annotation>
      <duration>276116</duration>
      <image>
        http://images.amazon.com/images/I/41bVr9y%2BKEL._SS400_.jpg
      </image>
      <info>http://example.com</info>
    </track>
  </trackList>
</playlist>
```

Figure 2.8: An example playlist serialised as XSPF

as other concepts around ordered listening of music such as skip counting. It is based on the Ordered List Ontology[11] and the Music Ontology [Raimond et al., 2007]. A graph rendering showing an example playlist's concept instance relationships is shown in Figure 2.9. The Playlist Ontology represents a still-maturing effort to create a common data model for playlists, across a diverse array of playlist creation, storage and playback environments. The need for such a common model is ever increasing as the ecology of applications around playlists and playlisting continues to grow and diversify.

2.7 Deployed Tools and Services

Over the past few years a number of software tools, including web services and applications, have been developed that concern playlists. Among these tools and services there is a great deal of variation in how playlists are involved, the interactivity model employed and various other factors. We consider them along 2 axes to form four quadrants, as can be seen in Figure 2.10. These four groups are labeled "construct non-social," "consume non-social," "consume social" and "construct social."

2.7.1 Construct Non-Social

There are a plethora of tools that are primarily considered *media players* that in addition to simply playing audio allow varying degrees of support for constructing and saving playlists. The most well known of these include: VLC[12], Apple iTunes[13], SongBird[14], WinAmp[15] and Microsoft's Windows Media Player[16] along with many more[17].

An emerging type of system in this category is best seen in *rush* [Baur et al., 2010]. Rush is designed as recommendation-based interactive model and visualisation technique to browse and listen to large collections of music. Beginning with a seed song, a set of recommendations is delivered, with one being selected by the user and added to the play queue. The process then repeats with the selected songs serving as the new seed for recommended songs. In this way a playlist is constructed manually, though using automatically selected subsets of the total database of songs.

[11] http://smiy.sourceforge.net/olo/spec/orderedlistontology.html
[12] http://www.videolan.org/vlc/
[13] http://www.apple.com/itunes/
[14] http://getsongbird.com/
[15] http://www.winamp.com/
[16] http://www.microsoft.com/windows/windowsmedia/default.mspx
[17] A fairly complete list of audio players can be found at http://en.wikipedia.org/wiki/Comparison_of_audio_player_software and http://en.wikipedia.org/wiki/Comparison_of_video_player_software. Note that any player that *can* play video will be on the second list and not the first, regardless of its audio capabilities.

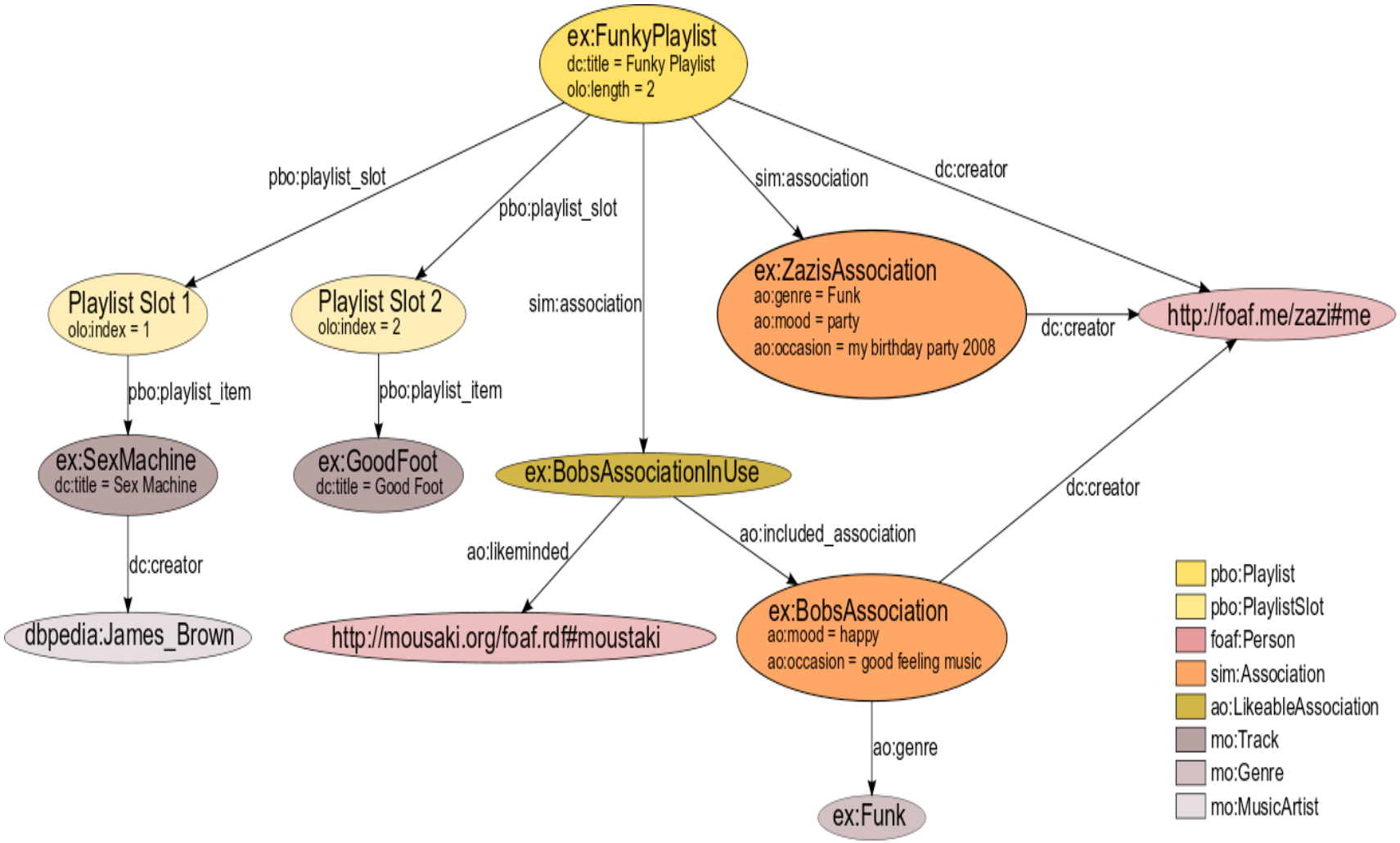


Figure 2.9: A relational graph rendering of an instance of the Playback Ontology, with classes colour-coded and labelled

Figure 2.10: The distribution of a selection of playlist creation, manipulation and listening tools represented by their icons or logos. Note Apple iTunes is both an construction and a consumption system, depending on use.

2.7.2 Consume Non-Social

The breadth of consumption non-social systems can be considered as local playlist creation assistance tools. That is they help a user to construct playlists using primarily locally available information, using a variety of techniques. One of the most common tools of this sort is a feature within Apple iTunes called "smart playlists" [Voida et al., 2005]. This feature allows a user to specify a number of rules or constraints on various kinds of metadata which are then used to create playlists automatically. A broad range of metadata can be used including common labels such as artist name, track title or tempo (in beats per minute or BPM) as well as behavioural statistics around the songs such as number of skips, times played or dated added to library. In many ways this system can allow for the automatic construction of playlists that are similar to the sort of playlists made by hobbyists [Cunningham et al., 2006], as discussed in Section 2.5. However, there are reasons to doubt this particular tool's popularity and use. In an informal survey of his readership[18], Paul Lamere looked into how prevalent the use of smart playlists are amongst the readers of his music technology blog[19]. The results of this survey, as of 9 September 2010, can be seen in Figure 2.11. At least among this sample, up take of these lists is not very high, though the reasons are not given.

Another example from Apple is the iTunes "Genius Mix", which takes your collection of music and divides it into 12 top-level genre groupings. A listener then selects which group to listen to, and songs are chosen at random

[18] http://musicmachinery.com/2010/07/30/do-you-use-smart-playlists/

[19] This blog is read by music technologist who are early adopters and as such this survey if anything represents an overstatement of the uptake of smart playlists.

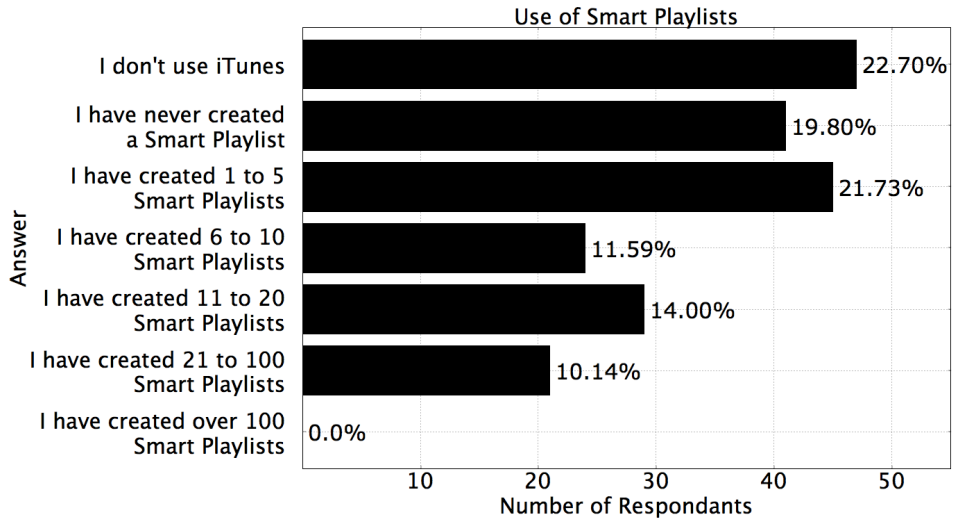


Figure 2.11: Smart playlists used per user, an informal survey

from within this set. While the method can be effective, it allows for artist repetition and lacks many user controls or transparency.

There are also a number of web-based services that offer a *personalized radio* service, using various metadata and content-based analytics to generate playlists based on textual queries or seed songs. Perhaps the most well known of these services (though not available outside the United States) is Pandora[20], which uses trained listeners to describe music using a large set of binary classifiers referred to as *music dna* [Clifford, 2007].

Another service along somewhat similar lines are MOG[21] and the mobile device application Moodagent[22]. Moodagent provides streaming playlists suited toward a user's specified ratio of 5 dimensions of 'mood': 'sensual,' 'tender,' 'happy,' 'angry' and 'tempo.' The user has the ability to adjust these parameters at any time and the playlist will change accordingly. A screen capture of the interface can be seen in Figure 2.12.

Van Gulik and Vignoli [2005] describe a method of automatically constructing playlists by drawing paths though semantically meaningful spaces, but removed from specific pieces of music. The dimensions can represent a wide variety of things, from publication related metadata such as year of release to metadata describing the content such as tempo. The tool then allows trajectories to be plotted based on these dimensions. Using the example dimensions one could specify a playlist that begins with high BPM in the 1970s,

[20] http://pandora.com
[21] http://mog.com
[22] http://www.moodagent.com/

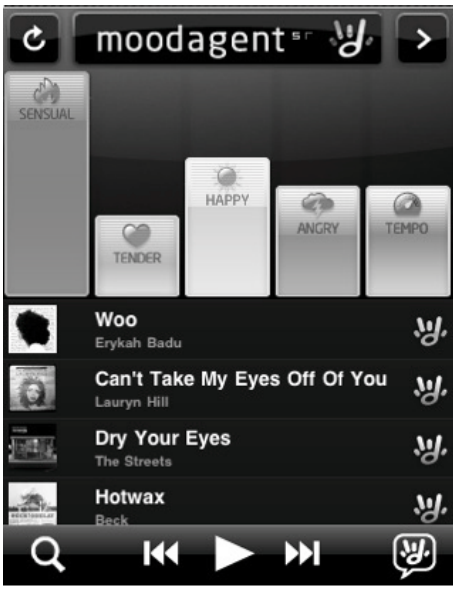


Figure 2.12: The user interface for the mobile application Moodagent

keeps the tempo constant but goes back toward the 1960s then gradually slows the tempo while increasing the release year to the early 1990s.

Continuing with the idea of drawing a playlist on a created space, Lillie [2008] provides a visualisation of a collection of music based on a two dimensional principle component analysis (PCA) reduction of a high dimensional feature space, presenting it as a map. This map can then be navigated to find single tracks or trajectories can be drawn trough it to create playlists. This work builds on Lamere and Eck [2007], which creates a three dimensional space using content-based timbral features. This created space is presented for browsing, and playlists can be laid out as paths through this space.

2.7.3 Consume Social

Consuming music is frequently a social affair. One of the most popular web-based services for playlist generation in a social context is Last.fm[23] and its radio player. Last.fm's core data is the *scrobbling* history of its users. Scrobbling is the act of uploading a user's listening activity as it's happening. Scrobbling can be performed in many music listening environments (including many mentioned in Section 2.7.1) and as a result last.fm has a very large database of user listening activity[24]. This listening activity is combined with collaborative filtering to create rich recommendations based on listening history [Goldberg et al., 1992]. It is also used as the basis for a radio player available from the

[23] http://last.fm

[24] According to Last.fm over 45,537,161,650 tracks have been scrobble since 2003 (http://www.last.fm/community as of 12 September 2010).

site[25]. Additionally there are a number of social features built into the user experience. These features include a dynamic structure of *friend* connections with other users and the ability to see what your friends (the users you are connected with) have been listening to recently. These features can be seen in the screen captures in Figure 2.13.

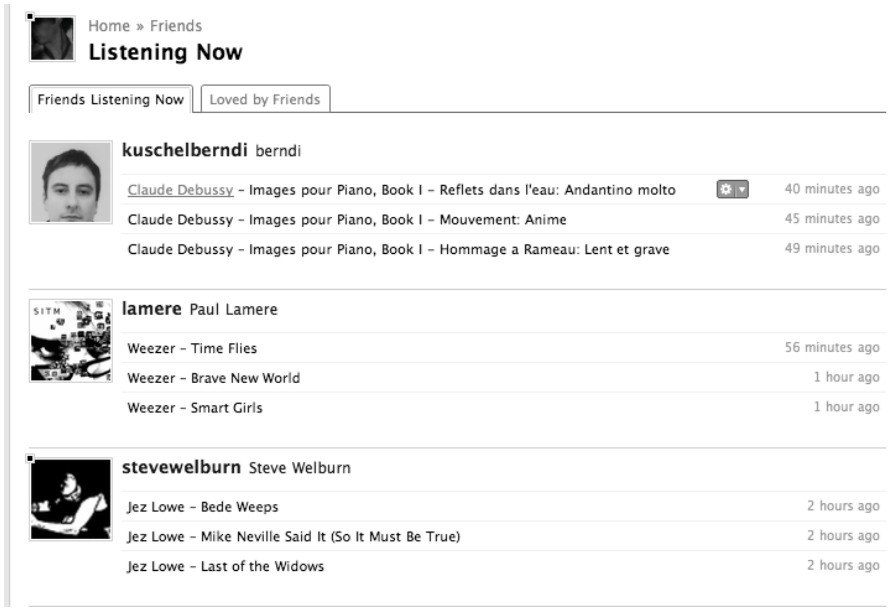
The internet radio station Radio Paradise[26] is also considered as an consumption social playlisting service. Since the playlists are constructed by a human DJ, the listeners themselves are not assembling them by hand, as they would be in the systems described in Sections 2.7.1 and 2.7.4, so they are considered consumption services from the listener's perspective. Further, internet audio streams such as Radio Paradise provide a single stream for all of their listeners, making for a social experience in common listening. Radio Paradise further expands this social experience via a song's comment stream. While comments are tied to individual songs, the particular transitions and play order is frequently discussed as the audience hears the same playlist and comments on a song's thread as it is played. Another significant factor affecting playlist construction in internet radio is a piece of United States regulation, Digital Millennium Copyright Act of 1998 (DMCA) (Sec 405.a.2).

The DMCA has a clearly defined set of rules that a radio station must follow in order to qualify for licensing fees as a broadcaster. If a broadcaster does not meet these rules, they must make direct arrangements with the copyright holder (usually a record label or publishing house), which is almost always prohibitively expensive [Harwood, 2004]. The rules the DMCA dictate in order to qualify as an internet broadcaster are as follows:

- In a single 3 hour period a station may broadcast:
  - No more than three songs from the same recording
  - No more than two songs in a row, from the same recording
  - No more than four songs from the same artist or anthology
  - No more than three songs in a row from the same artist or anthology
- Any program of songs that repeats immediately must be at least 3 hours long
- Any program that rebroadcasts:
  - Programs of less then one hour: no more than three times in a two week period

[25] http://www.last.fm/listen
[26] http://radioparadise.com

(a) The last.fm radio player



(b) A selection of friends recent listening

Figure 2.13: The radio player and social interaction on Last.fm

– Programs of greater than one hour: no more than four times in a two week period

There are a number of additional rules as well, though these relate to metadata transmission and do not affect song selection or order. While only applicable in the United States, given the large market the United States audience represents, this law's rules tend to be followed by most internet radio stations, including those based outside the United States.

In addition to internet-based radio stations, traditional terrestrial radio can also be considered as consumption social playlist generation sources; here too the listener is not actively participating in the construction of the playlist, regardless of whether it is created by a live DJ or an algorithm. Many of the norms of playlist construction come from terrestrial radio. A particularly interesting structure in radio programming is the concept of *dayparting* [Wells and Hakanen, 1997]. Dayparting is the idea of dividing the day into a set of (usually) 5 'dayparts': midnight to 6 am, 6 am - 10am, 10am-3pm, 3pm-7pm, and 7pm to midnight. Each daypart is then programed via a number of parameters and checks. There are daypart specific controls of items such as gender of artist, tempo, song intensity, mood and other various style controls. Typically there are *artist separation* controls, both globally and for individual artists. There is also *horizontal separation* of titles, preventing a song from being played at the same time of day, on consecutive days. A number of other rules enforce various norms such as *artist block*, preventing multiple songs from the same artist being played in a row, and additional *never-violate* and *preferred* rules. Time of day information has begun to be used in algorithmically generated music recommender systems, but this work is still developing [Herrera et al., 2010].

2.7.4 Construct Social

There are a number of web-based services and communities created around sharing playlists. In general, no assistance is offered in the construction of playlists, just encoding and sharing. Perhaps the oldest and most established of these communities is Art of the Mix[27]. This site is a community-driven forum for specifying and posting 'mixes.'

Cunningham et al. [2006] analyses the ways and particulars of playlist formation and suggestion on Art of the Mix. Additionally, a dataset of precrawled dataset of mixes[28] has been used as training data for automatic music similarity research [Berenzweig et al., 2004].

A number of services dealing with the exchange of playlists are much more

[27] http://artofthemix.org
[28] http://labrosa.ee.columbia.edu/projects/musicsim/aotm.html

focused on a search and retrieval user experience, rather than the bottom-up community of Art of the Mix. Notable examples in this group include fiql.com[29], Playlist.com[30] and mixpod[31]. These services allow you to search existing playlists via song metadata such as artist or track name, create your own playlists and share playlists.

However, since none of these services allow a user easy access to the music in the playlist, the playlists cannot be auditioned without finding and purchasing the music they contain. A group of web services that offer a solution to this issue are driven by Spotify API[32]. Spotify offers on demand streaming access to a very large database of music (in the order of 10^8 songs), under either a monthly subscription or free with inserted advertisements. Users can make playlists within Spotify and share them via a number of external sites[33]. Recently[34], Spotify added a number of social features directly into their application, allowing you to share songs and playlists among a group of friends.

Mixcloud[35] is a free social-networking platform organized around the exchange of long-form audio (*i.e.* digital audio files with a duration an order of magnitude longer than a standard song, typically between 30 and 90 minutes in length), principally music. It provides a means for DJs (aspiring and professional) to connect with the audience and into the Web. A similar service is mixlr[36] which focuses on adding social features to centralized multicasting. It supports live and recorded streams both mixed and unmixed. While its social connectivity is web-based, the broadcaster is a native application. This native application layer provides integration with common DJ tools.

An emerging service is that of setlist.fm[37]. This community driven site is a wiki for concert playlists. While still new, it is quickly becoming an excellent resource for song orderings from live performance.

2.7.5 Summary

There are many currently deployed systems that provide playlist-related services and features. These can take many forms ranging from music libraries with an ability to specify and store playback order, to radio stations with varying amounts of community support, to emerging persistent web services for

[29] http://fiql.com
[30] playlist.com
[31] http://mixpod.com
[32] http://developer.spotify.com/
[33] While there are far too many to exhaustively list them, notable examples include: http://sharemyplaylists.com/ and http://www.yourspotify.com/.
[34] http://www.spotify.com/uk/blog/archives/2010/04/27/the-next-generation/
[35] http://www.mixcloud.com
[36] http://www.mixlr.com
[37] http://setlist.fm

playback and sharing. While many of these systems continue to push what is possible, a complete, automatic solution has yet to be developed, but see Section 2.8.2 for research efforts toward that end.

2.8 Research Systems for Playlist Generation

There is a small body of work concerning the automatic generation of playlists. Most of these previous attempts at playlist generation rely on either textual metadata or content based features. There is current work attempting to merge these two data sources, which is discussed in Chapter 3. Further, the principal interface for specifying a playlist through most previous works is by example, via a single song that starts the playlist.

2.8.1 Human-Facilitating Systems

A number of research systems facilitate the creation and sharing of playlists, rather than automatically building playlists directly. Hayes and Cunningham [2000] detail an early collaborative filtering system in which users rate songs directly. Playlists are then built by finding similar (using Pearson's correlation coefficient) users via ratings profiles. These playlists are unordered sets delivered in a random order and once shown to the user they can be streamed, named, modified and shared.

Moving many of these ideas to a mobile device and introducing the idea of *continuous listening* is *Social Playlist* [Liu and Reimer, 2008]. This system was built with four design goals: music should help convey status information and implicit presence; music should help build interpersonal relationships; a good individual listening experience should be supported; and support of smooth continuous listening. Based on these goals a system was built and tested. The workflow for interacting with this system is as follows.

1. Members associate music from their personal library to their activities and locations.

2. To select the next song to be played, the system picks a random user and a song associated with that user's current state .

3. Music is streamed to each mobile device.

4. The device displays the current song and which user associated it their current state.

The generated playlist's music is broadcasted to each participating mobile device via a server. Everyone hears the same music in parallel, as if they were tuning into a common radio station. The system was field tested with a group

of five users over a period of two weeks. While users were generally happy with the interface, they reported hearing 30% - 50% "bad songs." This led the users to switch off the service and instead listen to music from their own libraries. In order to support continuous listening then, users must be given the ability to move away from the songs they strongly dislike (*i.e.* change stations).

Moving away from virtual meetings and non-place, the *Jukola* system provides everyone in a physical space a means to vote for the next song to be played in that space. It is installed in a space for public performance of recorded music such as a cafe or bar and consists of two different input devices, one main larger console and number of smaller devices distributed throughout the space [O'Hara et al., 2004]. Playlists are constructed on the fly, one song at a time, via a simple majority voting system across all devices. While any song known to the system can be voted for at the main console, only the *nominees*, with an additional random selection, can be voted on at the portable systems. Nominees for the portable systems are selected at the main station. Additionally, the system provides backend support for bands to upload material directly, via a web interface. Together the whole system represents a complete distribution chain for public performance of recorded music.

2.8.2 Fully-Automatic Systems

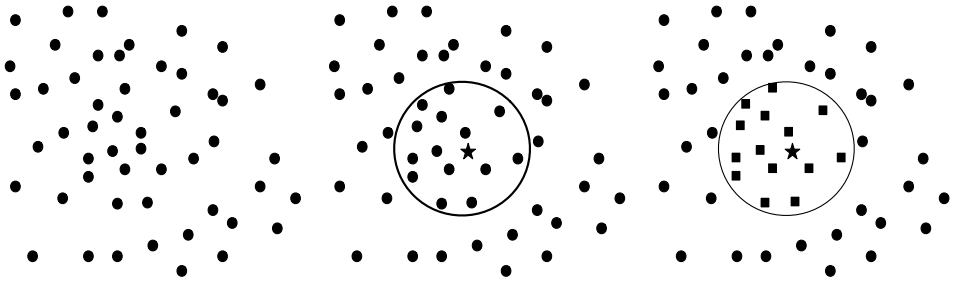
A number of fully automatic playlist generation systems have been examined in the literature. The basic premise in the simplest case is to take a distance measurement between pairs of songs in a collection; when given a seed song, return a list of its nearest neighbors as a playlist [Aucouturier and Pachet, 2002; Ragno et al., 2005]. In the generic this process is as follows.

1. Distances between songs are calculated, typically through the use of an extracted feature with a statistical average or dimensional reduction applied. Standard examples include Mel-frequency cepstral coefficients (MFCC) summarised using Gaussian mixture models (GMM). Distances between GMMs are calculated, usually via earth mover's distance or Kullback–Leibler divergence [Aucouturier and Pachet, 2004; Berenzweig et al., 2004; Pampalk, 2006].

2. A seed song and distance threshold are then selected. In some systems the distance threshold is selected directly whereas in others the threshold is selected indirectly based on the number of songs that will be within the radius of the threshold, to create a playlist of a given length of songs.

3. The playlist is created using all the songs inside the threshold radius.

This process is illustrated in Figure 2.14.



(a) An initial projection of songs, showing their distance from each other in a metric space.

(b) A seed song has been selected (now a star) and a radius of interest is shown.

(c) Finally, songs within the threshold radius will be included in the playlist (now squares) and outside this radius will not be used (unchanged shape).

Figure 2.14: The process of nearest neighbour unordered playlist creation

Logan [2002] describes a graph-based variant of this process. In this work a weighted graph is constructed using acoustic similarity based on MFCCs and summarised with a single Gaussian model. Playlists are then taken from this graph by the following means: a seed song and desired playlist length of N songs are chosen; the path of length N with the smallest total weight is selected as the playlist. This method was then compared the the nearest neighbors method with the results being shown to be fairly similar when evaluating using genre, artist and album labels. A selection of results from this work can be seen in Table 2.6.

In work fairly indicative of textual metadata playlist generation, Platt et al. [2002] uses Gaussian-process regression, a form of meta-training, to formulate a kernel based on text metadata hand entered describing a large corpus of albums. The metadata used is quite extensive; types and examples for each type can be seen in Table 2.7. The trained kernel is then used to generate Listener-to-Self type playlists that are shown to better predict a listener's playlists than a kernel which has been designed by hand. While this method is only lightly evaluated it shows promise. Like any system based purely on textual metadata, this system can only be as good as the metadata of the user's collection. A system such as this would not do well with a set of poorly formed, missing or incorrect metadata as the playlists generated are optimised around metadata, with no regard for the audio.

Another variant of the textual metadata case uses social relationships and common taste to make tailored playlists through collaborative filtering [Avesani et al., 2002; Hayes and Cunningham, 2004]. In this approach, by finding users with overlapping interests, recommendations can be made based on the strength

Relevance	Scheme	mean relevant songs in playlists		
		Size 5	Size 10	Size 20
Same Genre	Trajectory, 1	3.26	6.13	10.75
Same Artist		1.08	1.48	1.68
Same Album		0.89	1.11	1.22
Same Genre	Trajectory, 2	3.33	6.37	12.08
Same Artist		1.23	1.89	2.73
Same Album		1.01	1.49	2.00
Same Genre	Feedback	3.40	6.54	12.46
Same Artist		1.27	1.96	2.83
Same Album		1.05	1.54	2.07

Table 2.6: Relevance of playlists generated via two graph trajectories and k-NN, evaluated against genre, artist and album identity.

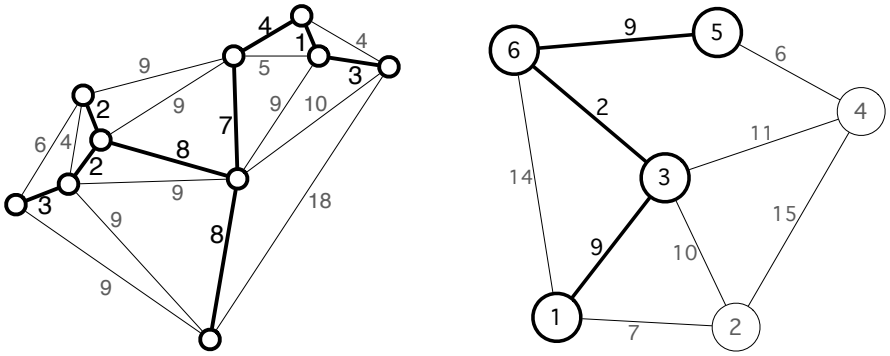
Metadata field	Example Values	N
Genre	Jazz, Reggae, Hip-Hop	30
Subgenre	Heavy Metal, I'm so sad and spaced out	572
Style	East Coast Rap, Gansta Rap, West Coast Rap	890
Mood	Dreamy, Fun, Angry	21
Rhythm Type	Straight, Swing, Disco	10
Rhythm Description	Frenetic, Funky, Lazy	13
Vocal Code	Instrumental, Male, Female, Duet	6

Table 2.7: Metadata fields and example valued from Platt et al. [2002]. N is the number of possible values for a field.

of feedback from others with similar interests who have already listened to unknown works.

Moving away from generation methods reliant on a single seed song, *Traveller's Sound Player* uses an approximate solution to the Travelling Salesman Problem (TSP) to generate *tours* through a collection of music [Knees et al., 2006]. The system uses a combination of content-based song and web-based artist similarity to generate a distance matrix. Here 'web-based artist similarity' is created by weighted terms found on the top 50 websites returned by a google search for the artist's name and the term 'music.' These artist similarities are used to create a Self-Organising Map (SOM) [Kohonen, 1990], clustering similar artists together. Every artist in the same SOM cluster then has their songs compared via an acoustic similarity measure, using MFCCs and GMMs, creating a song to song transition model for the entire collection such that the sum of the distances of neighbouring songs in minimal. An approximation of the TSP is used to find tours through the entire collection. This algorithm was then tested on two collections of about 3000 tracks each with a number of SOM cluster sizes to vary the portion of the collection compared via acoustic analysis. The resulting tours were then evaluated in terms of the frequency of neighbouring tracks changing genre, showing that for these collections a 5×5 SOM (25 clusters, each with about 120 audio tracks) performed best. This argues for a balanced approach between metadata sources encoding sociocultural factors (here the weighted terms from webpages) and acoustic features to form similarity measures.

As both Logan [2002] and Knees et al. [2006] make clear, many playlist-generating methods depend on graph-theoretic techniques for construction. In addition to the techniques of k-NN and TSP used in these papers, other techniques to generate playlists for various use cases can be taken from graph theoretic approaches. The minimum-spanning tree problem attempts to find a *tree* (*i.e.* a graph in which every pair of nodes is connected by exactly one simple path) with the minimum number of edges. In the case of *weighted edges*, where some edges represent a longer or shorter distance than others, the minimum spanning tree becomes nontrivial, finding not the minimum number of edges but the minimum sum of weights necessary to create a tree from a given graph [Graham and Hell, 1985]. The shortest-path problem is one of finding the path which is traverses the lowest total distance between two nodes in a graph [Dreyfus, 1969]. In cost networks this is also described as finding the 'cheapest' path, though the underlying algorithms in both cases are equivalent. These two graph problems are illustrated in Figure 2.15.

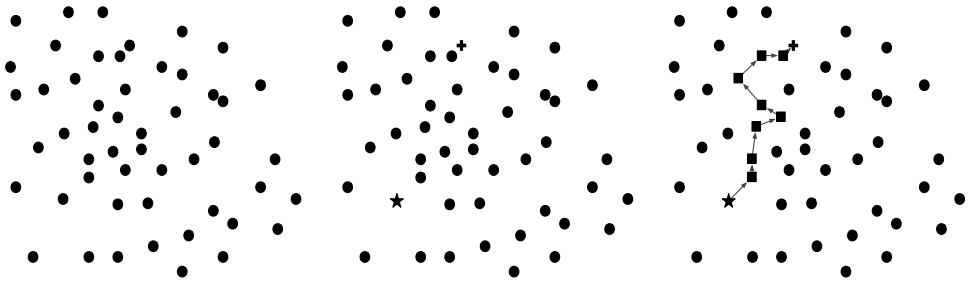


(a) A graph showing the minimum spanning tree in bold.

(b) A graph showing the shortest path from node 1 to node 5.

Figure 2.15: Example solutions to the 'shortest path' and 'minimum spanning tree' problems. The numbers next to each edge are the weight or cost of a given edge

In addition to modelling songs as nodes of a connected graph, it is common in automatic music similarity work to model the relation between songs as points in a low dimensional space. These low-dimensional spaces are typically created by extracting higher dimensional features and applying a dimensional reduction technique such as Multi-Dimensional Scaling (MDS) [Chen et al., 2008] or Principal Component Analysis (PCA) [Jolliffe, 2005]. Once the space has been created, there are a number of ways to create playlists to traverse them. While the kNN approach discussed earlier required only specifying a seed song, if a seed song and a final song are specified a path can be found to minimise the distance between any one transition. This technique can be seen in Figure 2.16.



(a) A number of songs mapped to a $2D$ space.

(b) The start (star) and end (cross) songs have been selected.

(c) A path is selected minimising the distance between any single song transition.

Figure 2.16: The generic process of finding a path through a map of songs with the start and end songs specified.

An example of work close to this idea is Flexer et al. [2008]. Rather than project distances onto some n dimensional space, KL divergence between gaussian summaries of MFCCs of song pairs is taken directly. Ratios of these direct measures replace a projected distance space. For each song i in the collection, the divergence is calculated between the start song s ($D_{KL}(i,s)$) and end song e ($D_{KL}(i,e)$). If the song is not within the closest $d\%$ of songs from either the start or end song, it is ignored. For all remaining songs compute the divergent ratio:

$$R(i) = \frac{D_{KL}(i,s)}{D_{KL}(i,e)} \tag{2.5}$$

The ideal position to find a song when moving from the s to e is then computed.

$$\hat{R}(j) = R(s) - j\frac{R(s) - R(e)}{p+1} \tag{2.6}$$

Where p is the desired number of songs in the list. Finally, songs that best meet the ideal positions are found (note that once a song has been selected it is removed from the candidate list):

$$S_j = \text{argmin}_{i=1,\cdots,m} |\hat{R}(j) = R(i)| \tag{2.7}$$

The resulting playlists were then evaluated both objectively and subjectively. For the evaluation playlists were generated using 50 randomly selected songs from each of 6 genres as start and end songs, with the playlist always ending in a different genre than it started in. Objective analysis tested to see if a playlist created from song A to song B contained songs of the same genre as A and B. Further, at the beginning (first third) of the playlist most of the songs should come from the genre of song A, at the end (last third) most of the songs should be drawn from the genre of B and in the middle third the songs should be a mix of the genres of both songs. Here the results varied, with some genres dominating and overwhelming others, though with playlists between two well defined (in timbre space) genres the result is quite good. The aggregated results for playlists from "Hip-Hop" to "Rock" and "Funk" to "Pop" are shown in Table 2.8. The subjective analysis had evaluators listen to all the songs in a playlist using a cross-platform media player. The evaluator would listen to the start and end songs, then the songs in the between, freely moving through the playlist. For each playlist the evaluator would then specify how many of the songs were felt to be "outliers" and whether the presented order was "apparent." The results of this evaluation can be seen in Table 2.9.

In Pampalk et al. [2005] a playlist generation method is described which relies on an acoustic distance to determine a song's nearest neighbours. This

	Hip-Hop	Reggae	Funk	Electronic	Pop	**Rock**
First third	**33**	5	2	15	8	38
Middle	**5**	1	2	7	4	**81**
Last third	2	0	3	4	2	**88**

(a) The aggregated results of playlists from "Hip-Hop" to "Rock."

	Hip-Hop	Reggae	**Funk**	Electronic	**Pop**	Rock
First third	19	3	**8**	28	13	29
Middle	17	4	**4**	20	**19**	36
Last third	12	3	4	22	**16**	42

(b) The aggregated results of playlists from "Funk" to "Pop."

Table 2.8: The aggregated genre labels of playlists going from "Hip-Hop" to "Rock" and "Funk" to "Pop." The bold portions indicate the expected result based on the genre labels of the start and end songs.

Genres		**number of outliers**	**order apparent**		
from	**to**		**yes**	**somewhat**	**no**
Hip Hop	Reggae	4.7		x	xx
Hip Hop	Funk	1.7	xx	x	
Hip Hop	Elect	1.3	xxx		
Hip Hop	Pop	2.7		xx	x
Hip Hop	Rock	0	xxx		
Reggae	Funk	0.7	xx	x	
Reggae	Elect	1.3	xxx		
Reggae	Pop	1.3	xxx		
Reggae	Rock	0.3	xx		x
Funk	Elect	1.0	xx	x	
Funk	Pop	1.7	xx		x
Funk	Rock	0	xx	x	
Elect	Pop	0	xxx		
Elect	Rock	0	xx	x	
Pop	Rock	0	xxx		
	Average	1.1	71.1%	17.8%	11.1%

Table 2.9: Subjective evaluation from Flexer et al. [2008], each x represents a third of response.

nearest neighbor list is presented as the user's initial playlist, which is then steerable via the use of a skip button. The skip button allows the recommendations to get better over time. These playlists are roughly analogous to a playlist somewhere in between a Peer to Peer playlist and the Expert to Listener case in that an expert is created over time by combining user feedback and similar user preference.

2.8.3 Evaluation

The evaluation of a playlist generation system presents a number of challenges due to the particulars of the task. A few evaluation techniques have been discussed in previous sections alongside their use; due to the importance and difficulty in evaluation, a survey of approaches is presented, along with the application of some objective analysis techniques to available services' playlists.

2.8.3.1 Subjective Evaluation

Perhaps the most obvious evaluation technique for a music informatics technique is controlled direct audition (e.g. listening tests). One of the most complete examples of listening tests for playlist evaluation was conducted in Pauws and Eggen [2002]. Direct audition was used to test five hypotheses with regard to the authors' playlist generating system, *Personalized Automatic Track Selection* (PATS):

1. Playlists compiled by PATS contain more preferred songs than randomly assembled playlists, irrespective of a given context-of-use.

2. Similarly, PATS playlists are rated higher than randomly assembled playlists, irrespective of a given context-of-use.

3. Successive playlists compiled by PATS contain an increasing number of preferred songs.

4. By extension, successive PATS playlists are successively rated higher.

5. Successive playlists compiled by PATS contain more distinct and preferred songs than randomly assembled playlists.

The evaluation had 20 participants, 17 male and 3 female. Each participant attended a total of eight sessions over four days. For each session, a user would choose a seed song for preselected *context-of-use*, or real-world environment in which the music is meant to be heard (i.e. dance party, a romantic evening or traveling in a car). Over the duration of the tests, two contexts-of-use were used across four sessions for each participant. After a seed song was selected, two playlists of 11 songs each were created, one using the PATS system and another

through random selection. A one minute excerpt from each song in each playlist is then played for the listener. For each of these excerpts the listener assigns a judgement of suitability of a given song and then rates the entire playlist for overall fitness. These two assertions are used to create three measurements: *precision*, the proportion of songs in a playlist deemed as suiting the given context-of-use; *coverage*, the cumulative number of songs that suited the given context-of-use and that were not already present in the previous playlists, and *ratings score*, the participants' direct rating from 0 to 10, with 10 being the highest, of each playlist. Overall, using these metrics the PATS system was shown to significantly outperform the random selection of playlists. However, this difference was mostly flat, not showing the desired improvement over time within the given metrics. These results are directly usable as they are human-driven. This comes at a cost, as evidenced in the small sample and use of one-minute excerpts rather than whole songs.

In an effort to investigate the quality added by various orderings of music and people's ability to discriminate between the sources of playlists, Paul Lamere conducted a web-based survey[38] comparing playlists i) from the internet radio station Radio Paradise, ii) automatically generated via a hidden Markov model (HMM) trained on the Radio Paradise playlist, and iii) random selection from the set of songs occurring anywhere in these playlists. The playlists from Radio Paradise represent 18 months of continuous logs from January 2007 to July 2008. This gives 45,283 playlists containing 6,325 unique tracks from 4,094 albums by 1,971 artists. The average length of these playlists is 4.3 songs.

For each evaluation, a listener is presented with 12 ordered playlists with the ability to play the songs in the list in any order and as many times as the listener would like. After listening to the songs in a playlist the listener rates the playlist from 1 to 5 with 5 being the highest and attempts to guess whether the playlist was generated by a human expert (Radio Paradise DJ), algorithm (HMM), or via random selection. The survey rating for each generator type, both the assumed type and the actual type are shown in Table 2.10 and the confusion matrix across the playlist creation types is shown in Table 2.11. The results from this survey illustrate an assumed order of fitness from listeners based on the generator source. Table 2.10 shows that people expect their favourite playlists to have been made by human experts with those rated slightly lower coming from an algorithm and the least favourable being created by random selection. However, as is evident in the right half of the same portion, when the evaluators ratings are grouped by the correct labels, this

[38]The survey can be taken at http://labs.echonest.com/PlaylistSurvey/ with extended information at http://musicmachinery.com/2010/06/18/the-playlist-survey/.

	Guessed labels		True labels	
label	rating	count	rating	count
Human Expert	3.33	368	2.49	400
Algorithm	2.76	373	2.63	403
Random	2.08	343	2.64	386

Table 2.10: Aggregate ratings from Lamere's playlist survey. Note that the counts for guessed label ratings are a bit lower as some of those taking the survey did not guess the creation mechanism but did rate a given playlist.

		Guessed label		
		Human Expert	Algorithm	Random
true label	Human Expert	121	124	112
	Algorithm	122	126	123
	Random	125	121	107

Table 2.11: The confusion matrix between actual generator source and the listeners' assumed generator source for the three kinds of generated playlists in Lamere's playlist survey.

ordering falls away, with all three sources being within standard error of each other. This shows that in spite of preconceived notions the evaluators might have, all three methods of ordering perform about as well. From this data it appears that listeners expect human experts to outperform algorithms which should in turn outperform random choice, though these listeners' own rating betray their expectations. Note that this result is almost certainly tied to the fact that all three of these generation methods selected and ordered songs from the same collection set, as created by the human-expert (the DJ at Radio Paradise, Bill Goldsmith). So in some ways the situation is similar to the situation described in Section 2.5.2 when looking into shuffle ordering: with a coherent set of songs, intentional ordering becomes less necessary and random order can lead to serendipity.

Subjective evaluation, while costly in both time and resources, is the only way to determine the fitness of a playlist generation method for human listeners. It is clearly not a panacea, since when directly asked, people are not the best discriminators, highlighting the necessity of consistently good experimental design[39].

[39]That is to say, it is better to ask human subjects 'Do you like this (for this situation)?' rather than 'Where did this come from?' or 'Who made this?' When doing subjective experimentation, ask subjective questions

source:	RP	Musicmobs	AotM	Pandora
Playlists	45,283	1,736	29,164	94
Unique Artists	1,971	19,113	48,169	556
Unique Tracks	6,325	93,931	218,261	908
Mean Number of Songs	4.3	100	20	11
Duplicate Artists	0.3%	79%	49%	48%
Consecutive Same Artist	0.3%	60%	20%	5%

Table 2.12: Basic statistics for the playlist used for objective evaluation, by source. Note that the Pandora playlists were gathered by directly using the service, leading to a small sample from 44 separate stations. **RP** is internet radio station Radio Paradise and **AotM** is the website art of the mix.

2.8.3.2 Objective Evaluation

One of the most common ways to analyse playlists in an objective way involves measurements of similarity across the member songs by looking at the occurrence of identical genre, artist or album labels [Flexer et al., 2008; Logan, 2002] or the likelihood of these labels to change across neighbouring members of a playlist [Knees et al., 2006]. While this gives a reasonable measure of homogeneity across a playlist, in light of the elements of a good playlist discussed in Section 2.5, it is unclear if a measure of homogeneity corresponds to fitness. A bag of similar tracks is not the same as a suitable playlist. In order to have a better sense of possible objective measures we introduce the notions of *tag diversity* and *playlist cohesion* and apply them to playlists from existing services [Fields and Lamere, 2010]. These measures were used on playlists gathered from a number of sources; some basic statistics about the playlists are shown in Table 2.12.

Tag diversity is a measure of the song homogeneity in a playlist. This is a ratio of the number of *unique tags* describing the artists of the tracks in a playlist versus the total number of artist tags in a playlist. The artist tags were retrieved from Last.fm in 2007 and were then filtered so only tags that were applied by multiple people were retained Eck et al. [2007]. The average tag diversity of the playlists from each of the services being examined along with the tag diversity of randomly generated lists from the services' set of tracks used to create the lists is shown in Table 2.13. While similar in intent to the previously-mentioned bag of similar tracks evaluation methods, by using social tags the precision and complexity encoded in the measure is considerably better then standard metadata such as genre labels.

We see considerable variation in the homogeneity of playlists produced by

	Actual		Random	
source	mean	whole	mean	whole
Radio Paradise	0.74	0.13	0.75	0.13
Musicmobs	0.29	0.18	0.51	0.13
Art of the mix	0.48	0.17	0.61	0.11
Pandora	0.44	0.20	0.64	0.19

Table 2.13: Artist tag diversity of playlists retrieved from various sources, comparing both the *mean* for all playlists and the diversity of the *whole* set of songs for each source

these services, as might be expected given the diverse use cases they aim to meet. Radio Paradise, having not just the highest diversity but maintaining that diversity through the randomisation of the playlists, may give a window into the song selection process used in the playlist turing test. Also of note is Musicmobs, having both the longest playlists (mean is 100) and the lowest diversity. This bears out in qualitative examination of the data, where many of the playlist are on topics such as "artist retrospective" giving a curated tour through an artist's discography.

Playlist cohesion is a measure of the average step size between songs across a playlist. In order to compute the cohesion of a given playlist, we first create a weighted connected graph of an item space. The level of representation of the nodes in the graph determines the cohesion level examined. For any ordered playlist, find the shortest path which traverses the sequence of songs in the playlists. The average step size (*i.e.* total weight) between playlist items is the the cohesion of a playlist. An example of cohesion measured using an arbitrary graph is shown in Figure 2.17

To calculate cohesion on the collected playlists, we use two different weighted graphs, generating two cohesion measures. The first graph is of artist relationships from MusicBrainz[40]. Here the nodes are artists and the edges represent an artist-to-artist relationship. We weight these relationships as shown in Table 2.14. These weights attempt to encode the relative strength of various kinds of artist-to-artist interactions. These particular weightings were originally used in the Six Degrees of Black Sabbath[41] graph walking project. The second graph also represents artists as nodes. The edges are created between similar artists according to The Echo Nest's comprehensive artist similarity[42].

[40] http://musicbrainz.org
[41] http://labs.echonest.com/SixDegrees/
[42] http://developer.echonest.com/docs/v4/artist.html

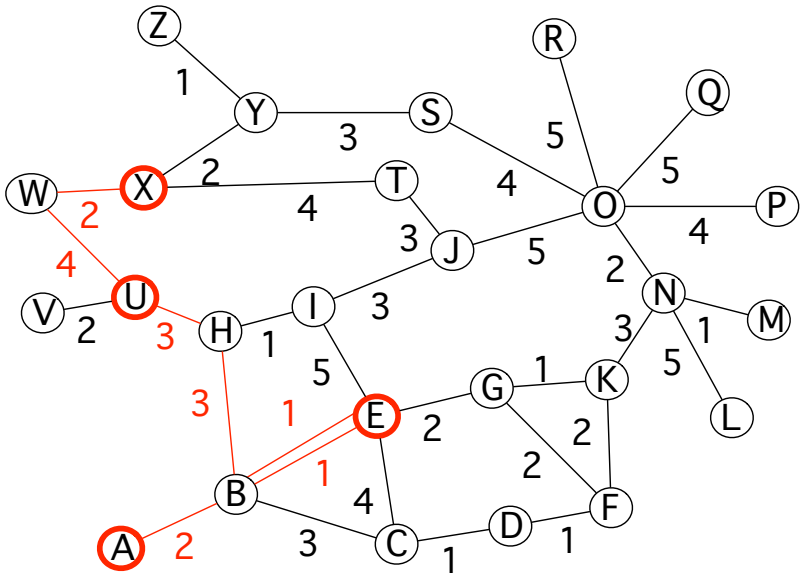


Figure 2.17: This graph shows the playlist $[A, E, U, X]$ with the shortest path between each node $[3, 7, 6]$ for a cohesion of $\frac{16}{3} = 5.33$

edge type	**weight**
Is Person	1
Member of band	10
Married	20
Performed with	100
Composed	250
Remixed	500
Edited Liner Notes	1000

Table 2.14: Weights assigned to the various artist-to-artist relationships in the MusicBrainz artist graph as used to calculate playlist cohesion.

	MusicBrainz		Echo Nest Sim	
Source	Cohesion	Global weight	Cohesion	Global weight
Radio Paradise	0.08	0.06	2.27	1.0
Pandora	0.11	0.12	1.57	1.4
MusicMobs	0.13	0.10	2.71	1.7
Art of the mix	0.14	0.10	3.02	1.4
Random (RP)	0.27	0.22	4.02	1.2
Random (graph)	0.39	0.45	7.89	1.7
Random (AotM)	0.56	0.19	7.00	1.1

Table 2.15: Playlist cohesion from both the MusicBrainz artist graph and the Echonest artist similarity graph.

This similarity is also used to weight the edges. In order to construct the path of a playlist on these artist graphs, we use the artist credited with each song in a playlist, creating an ordered list of artists. Between each pair of neighbouring artists, cohesion is calculated, with these individual cohesions averaged across a playlist. The resulting average cohesion for each playlist source on both graphs is shown in Table 2.15. Both the MusicBrainz and Echonest graphs show Pandora's playlists as being the most cohesive and one of the random generators being the least cohesive. This is unsurprising as Pandora's artist radio playlists typically feature a very homogenous selection of artists by design and the random graphs are selecting equally from across the entire dataset, giving a heterogeneous selection of artists in the produced playlists.

2.9 Discussion

We have proposed the consideration of a playlist as a specialised form of a recommender system. To inform this framing a brief history of playlist creation was then presented. This history exposed the dependency of playlist creation on member songs' relationships to one another, thus their similarity. Therefore, a discussion of music similarity followed, focused on current state of the art in automatic music similarity, as represented by the MIREX 2010 AMS task. Using this background, we surveyed currently deployed tools of throughout the ecosystem of playlist generation, from audio library management software to web-based radio and music-centric social networks. The survey then turned to research systems which are similarly varied, though they tend to focus more on content and less on users. Finally, we reviewed current methods of evaluation along with some proposed novel methods of evaluation, which were used and shown to be of interest when applied to currently deployed systems.

One of the most critical insights given all of this is the dependency between music similarity (or more broadly, the various relationships between pieces of music) and the construction of playlists. When we examine the state of the art in automatic music similarity, it is apparent that the focus is principally on content-based similarity. Yet recent work shows that a much of a person's musical taste is driven not by information with a digital audio recording of a piece of music, but rather social-cultural factors [Laplante, 2010]. This is not to say that content-based methods should be abandoned; rather, this speaks to a need to hybridize automatic similarity methods to include socio-cultural information, while maintaining a working understanding of content-based similarity, especially when determining final song order.

It is also impossible to ignore the rapidly-changing circumstances of music distribution, both commercial and amateur. Web-based streaming services, whether supported by advertisement or subscription, such as Spotify and Mog are rapidly changing assumptions about the ownership of recorded music. These services are becoming more ubiquitous with the continued growth on broadband on mobile phones, leading to constant access to near-complete collections of commercial music. As services of this type stabilise and become ubiquitous, the line between music that is *yours* and music *you want to listen to* fades. With it, what it is to recommend a piece of music shifts from a suggestion to purchase to a suggestion to listen. As a result, the playlist will cease to be *a* music recommender, but rather *the* music recommender. It is in this context that we will present a more socially aware hybrid similarity space, a deployed web-radio system using playlist from this space and an improved metric to describe and compare playlists, given the importance of social context.

Chapter 3

Multimodal Social Network Analysis

> "We can no longer maintain any distinction between music and discourse about music, between the supposed object of analysis and the terms of analysis."
> –Bruce Horner, *Discourse*, 1999

3.1 Introduction

As more freely-available audio content continues to become accessible, listeners require more sophisticated tools to aid them in the discovery and organization of new music that they will find enjoyable. This need, along with the advent of Web-based social networks and the increasing accuracy of signal-based music information retrieval, has created an opportunity to exploit both social relationships and acoustic similarity in recommender and discovery systems. However, current systems have tended to use one of these techniques in isolation. In our view, combining these techniques provides a means to improving the understanding of the complex relationship between song objects that ultimately will lead to improved song recommendation. The most obvious way to do that is to base recommendations on more information than is provided by a single distance measure between songs. This would allow the production of systems capable of mediating content-based recommendations with given social connections and the construction of socially structured playlists.

Motivated by this, we examine the Myspace artist network. Though there are a number of music-oriented social-networking websites (*e.g.* Soundcloud[1], Jamendo[2], etc.), Myspace[3] is the *de facto* standard for web-based music artist promotion. Although exact figures are not made public, recent estimates suggest there are over 8 million artist pages[4] on Myspace.

[1] http://www.soundcloud.com/
[2] http://www.jamendo.com/
[3] http://www.myspace.com/
[4] http://techradar1.wordpress.com/2008/01/11/facebookmyspace-statistics/

The Myspace social network, like most social networks, is based upon undirected relational links between *friends* designating some kind of association. A link is created when a user makes a request, and another accepts the request to become friends; both users are then friends and an undirected link is established. Within each Myspace user's friends there is a subset of between 8 and 40 *top friends*. While generic friends are mutually confirmed, individual users unilaterally elevate friends to become top friends from the generic friends set. It is these top friends which are displayed in a user's profile page – other friends require one or more *click-throughs* to access them. In addition, any user can declare themselves as an *artist* which requires them to provide audio or video content. In our work we concern ourselves only with these artist users to limit the scope of our investigation to only those nodes on the graph that have audio content. For the purpose of this paper, *artist* and *artist page* are used interchangeably to refer to the collection of media and social relationships found at a specific Myspace page residing in Myspace's artist subnetwork.

Social networks present a way for nearly anyone to distribute their own media. As a result, there is an ever-larger amount of available music from an ever-increasing array of artists.

1. Given that this music is published within a relational space, how can we best use all of the available information to discover new music?

2. Can both social metadata and content-based comparisons be exploited to improve discovery of new material?

3. Can this crowd-sourced tangle of social networking ties provide insights into the dynamics of popular music?

4. Does the structure of a network of artists have any relevance to music-related studies such as music recommendation or musicology?

To work towards answering the questions posed above, we explore a subset of the artist network and consider only their top friend connections. We analyse this network and measures of acoustic distance (according to techniques using content-based analysis, which we will describe in the next section) between these artists. Furthermore, we identify communities of artists based on the Myspace network topology and attempt to relate these community structures to musical genre. Finally, we present a prototype system of music playlist generation, with particular attention paid to the means for its evaluation.

Immediately following this section is a review of relevant literature from complex network theory and signal-based music analysis. Next, a detailed discussion of our data acquisition methods to build a sampled data set is presented.

This is followed by a broad analysis of this data set in Section 3.3. The initial experiments into the relationship between the social connectivity and the acoustic feature space are described and their results presented and discussed in Section 3.4. Finally, the implications of this chapter are explored in Section 3.5 followed by connections for playlist creation in Section 3.6.

3.2 Networks and Audio

We begin with a discussion of existing tools for the analysis and manipulation of networks in Section 3.2.1. This section covers complex network analysis, network flow analysis, particular issues pertaining to networks of musicians and community structure. In Section 3.2.2 we examine highlights of past work in audio content-based music similarity.

3.2.1 Existing Tools for Networks

3.2.1.1 Complex Networks

Complex network theory deals with the structure of relationships in complex systems. Using the tools of graph theory and statistical mechanics, physicists have developed models and metrics for describing a diverse set of real-world networks – including social networks, academic citation networks, biological protein networks, and the World-Wide Web. It has been shown that these diverse networks often exhibit several unifying characteristics such as small-worldness[5], scale-free degree distributions, and community structure [Newman, 2003].

A given network G is described by a set N of *nodes* connected by a set E of *edges*. Each edge is defined by the pair (i, j) of nodes it connects. This pair of nodes are *neighbours*. If the edges imply directionality, *i.e.* $(i, j) \neq (j, i)$, the network is a *directed network*. Otherwise, it is an *undirected network*. Since we are dealing primarily with the *top friends* sub-network of myspace artists, in this paper all edges are directed unless otherwise stated. In some graphs each edge (i, j) will have an associated label $w(i, j)$ called the *weight*. This weight is sometimes thought of as the cost of traversing an edge, or an edge's resistance. The number of edges incident to a node i is the *degree* k_i. In a directed network there will be an *indegree* k_i^{in} and an *outdegree* k_i^{out} corresponding to the number of edges pointing into the node and away from the node respectively. The *geodesic* d_{ij} is the shortest path distance from i to j in number of edges traversed. The largest geodesic distance in a network is known as the *diameter*.

We will discuss some of the characteristics of the Myspace artist network

[5] A network is considered to exhibit small-worldness when most nodes are not neighbours yet can be reached through a relatively small number of connected intermediary nodes Watts [1999].

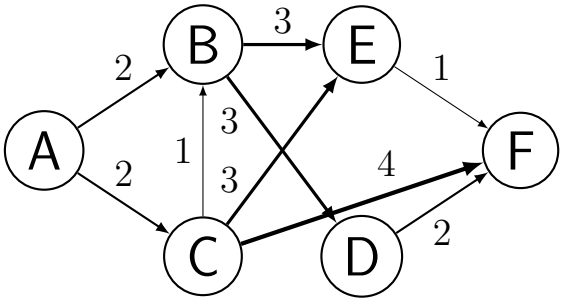


Figure 3.1: A simple flow network with directed weighted edges. Edge width is representative of node capacity, which is also labelled on each edge. Treating node A as the source and node F as the sink, the maximum flow is 4.

in 3.3.2. For a more in-depth discussion of complex network-analysis techniques the reader is referred to [Costa et al., 2007; Newman, 2003].

3.2.1.2 Network Flow Analysis

The basic premise in network flow analysis is to examine a network's set of nodes as sources and sinks of some kind of *traffic* [Ahuja et al., 1993]. Typically, though not exclusively, flow networks are directed, weighted graphs. Many useful measures for determining the density of edge connectivity between sources and sinks can be found in this space [Nagamochi and Ibaraki, 1992]. One of the most common among them is the maximum flow, which is a means of measuring the maximum capacity for fluid to flow between a source node to a sink node or, equivalently, the smallest sum of edge weights the edge of which must be *cut* from the network to create exactly two subgraphs, one containing the source node and one containing the sink node. This equivalence is the maximum flow/minimum cut theorem [Elias et al., 1956]. If the edges in a graph are unweighted, this value is also equivalent to the number of paths from the source to the sink which share no common edges. Mature algorithms, incorporating a number of optimization strategies, are available for computing the maximum flow between nodes [Ahuja et al., 1993; Goldberg and Tarjan, 1988].

An example of Maximum Flow can be seen on the network in figure 3.1. The narrowest flow capacity from node a to node f are the edges (a, b) and (a, c), where $w(a, b) + w(a, c) = 4$. The maximum flow can simply be found by taking the sum of the magnitude of each edge in the minimum cut set.

The few examples of network flow analysis being applied in music informatics deal primarily with constructing playlists using segments of a complete solution to the Traveling Salesman Problem [Knees et al., 2006]. Others use exhaustive and explicit textual metadata without comparisons to content-based metrics [Alghoniemy and Tewfik, 2001].

3.2.1.3 Musician Networks

Networks of musicians have been studied in the context of complex network theory – typically viewing the artists as nodes in the network and using either collaboration, influence, or similarity to define network edges. These networks of musicians exhibit many of the properties expected in social networks [Cano et al., 2006; Gleiser and Danon, 2003; Park et al., 2007]. However, these studies all examine networks created by experts (*e.g.* All Music Guide[6]) or via algorithmic means (*e.g.* Last.fm[7]) as opposed to the artists themselves, as is seen in Myspace and other similar networks. Networks of music listeners and of listeners connected to artists have also been studied [Anglade et al., 2007; Lambiotte and Ausloos, 2006].

3.2.1.4 Community Structure

Recently, as more data-heavy complex networks have been created across many domains, there has been a significant amount of interest in algorithms for detecting community structures in these networks. These algorithms are meant to find dense subgraphs (communities) in a larger sparse graph. More formally, the goal is to find a partition $\mathcal{P} = \{C_1, \ldots, C_c\}$ of the nodes in graph G such that the proportion of edges inside C_k is high compared to the proportion of edges between C_k and other partitions.

Because our network sample is moderately large, we restrict our analysis to use more scalable community detection algorithms. We make use of the greedy modularity optimization algorithm [Clauset et al., 2004] and the walktrap algorithm [Pons and Latapy, 2005]. These algorithms are described in detail in Section 3.3.3.

3.2.2 Content-Based Music Analysis

Many methods have been explored for content-based music analysis, attempting to characterising a music signal by its timbre, harmony, rhythm, or structure. One of the most widely used methods is the application of Mel-frequency cepstral coefficients to the modeling of timbre [Logan, 2000]. While a number of other spectral features have been used with success [Casey et al., 2008b], when used in combination with various statistical techniques MFCCs have been successfully applied to music similarity and genre classification tasks [Berenzweig et al., 2004; Logan and Salomon, 2001; Pampalk, 2006; Tzanetakis, 2007].

A simple and prevalent means to move from the high dimensional space of MFCCs to single similarity measure is to calculate the mean and covariance of each coefficient across an entire song and take the Euclidean distance between

[6] http://www.allmusic.com/
[7] http://www.lastfm.com/

these mean and covariance sets [Tzanetakis, 2007]. In the Music Information Retrieval Evaluation eXchange (MIREX) [Downie, 2006, 2008] competitions of both 2007[8] and 2009[9], this method, as employed by the Marsyas software suite, was shown to do a reasonable job of approximating human judgements of content-based similarity. A slightly more complex approach for computing timbre-based similarity between two songs or collections of songs creates Gaussian Mixture Models (GMM) describing the MFCCs and comparing the GMMs using a statistical distance measure. Often the Earth Mover's Distance (EMD) is the distance measure used for this purpose [Aucouturier and Pachet, 2004; Pampalk, 2006]. The EMD algorithm finds the minimum work required to transform one distribution into another. While the EMD-GMM approach models distance better than a simple Euclidean distance between averages of feature values, the simpler method may be sufficient and is considerably less computationally complex.

3.2.3 Measuring Independence Between Distributions

When comparing social and acoustic similarity in this work, in addition to examining linear correlation via Pearson correlation, we will also find the mutual information contained across the social and acoustic similarity distributions. Taken from information theory, mutual information is the amount of dependence (usually measured in bits) that one distribution has on another [Steuer et al., 2002, for example]. Given two distributions X and Y, Mutual information $I(X;Y)$ can be defined as

$$I(X;Y) = H(X) - H(X|Y) \tag{3.1}$$

where $H(X)$ is the marginal entropy of the joint distributions X and Y and $H(X|Y)$ is the conditional entropy of X given Y.

All mutual information and related entropy calculations in this work are calculated using pyentropy[10], a python library for performing information theoretic analysis on data distributions [Ince et al., 2009].

3.3 Data Set Acquisition and Analysis

Now that we have a foundational understanding of complex networks, we need to gather our data set. For reasons we discuss shortly it is not feasible to capture the entire Myspace artist network, we therefore take a sample which

[8] see http://www.music-ir.org/mirex/2007/index.php/Audio_Music_Similarity_and_Retrieval_Results entry by G. Tzanetakis

[9] see http://www.music-ir.org/mirex/2009/index.php/Audio_Music_Similarity_and_Retrieval_Results entry by G. Tzanetakis

[10] http://code.google.com/p/pyentropy/

we show to be representative. In this section, we report on our sampling of the Myspace network, describing our method in Section 3.3.1 and properties of this sample in Section 3.3.2. In order to examine the topography of our sample and the distribution of connectivity within the sample, we describe our methods for detecting community structure in Section 3.3.3.

3.3.1 Sampling Myspace

The Myspace social network presents a variety of challenges. Firstly, its size prohibits analysing the graph in its entirety, even when considering only the artist pages: therefore we sample a small yet sufficient portion of the network. Secondly, the Myspace social network is filled with noisy data – plagued by spammers and orphaned accounts: we limit the scope of our sampling in a way that minimizes this noise. Finally, there currently is no published interface for easily collecting the network data from Myspace. Our data is collected using web crawling and HTML document scraping techniques[11].

3.3.1.1 Artist Pages

It is important to note we are only concerned with a subset of the Myspace social network – the Myspace *artist* network. Myspace artist pages are different from standard Myspace pages in that they include a distinct audio player application containing material uploaded by that user. Standard practice (and a requirement of the End User License Agreement) is that this material has been generated by this user. We therefore use the presence or absence of this player to determine whether or not a given page is an artist page where, as stated in Section 3.1, *artist page* is used to refer to the collection of social links and audio material assumed to be generated by the same person or group of people.

A Myspace page will include a top friends list. This is a hyperlinked list of other Myspace accounts explicitly specified by the user and, unlike generic friends, need not be a reciprocal relationship. The top friends list is limited in length with a maximum length of 40 friends (the default length is 16 friends). In constructing our sampled artist network, we use the top friends list to create a set of directed edges between artists. Only top friends who also have artist pages are added to the sampled network; standard Myspace pages are ignored. We also ignore the remainder of the friends list (*i.e.* friends that are not specified by the user as top friends), assuming these relationships are not as relevant. Our sampling method is based on the assumption that artists specified as top friends have some meaningful musical connection for the user

[11] Myspace scraping is done using tools from the MyPySpace project available at http://mypyspace.sorceforge.net

– whether through collaboration, stylistic similarity, friendship, or artistic influence. This artificially limits the outdegree of each node in such a way as to only track social connections that have been selected by the artist to stand out, beyond the self-promoting noise of their complete friend list. Further, it is also a practical reduction as top friends can be scraped from the same single HTML document as all the other artist metadata. 50 friends are displayed per page, so gathering a full friend list would require $\frac{N}{50}$ pages to be scraped[12], significantly increasing the number of page requests required to sample the same number of artists.

In addition to these social connections, we also gather metadata about each artist. This metadata includes: the name of the artist, the number of page views, and genre labels associated with the artists. The audio files associated with each artist page in the sampled network are also collected for feature extraction. Note that genre tags collected are at the level of artists, rather than audio files; therefore all audio files associated with that artist will have the same genre labels applied (see Section 3.4.3).

3.3.1.2 Snowball Sampling

There are several network sampling methods; however, for the networks like the Myspace artist network, snowball sampling is the most appropriate method [Ahn et al., 2007; Lee et al., 2006]. In this method, the sample begins with a seed node (artist page), then the seed node's neighbours (top friends), then the neighbours' neighbours, are added to the sample. This breadth-first sampling is continued until the fraction of nodes in the sample reaches the target or *sampling ratio*. Here, we randomly select a seed artist[13] and collect all artist nodes within 6 edges to collect 15,478 nodes. If the size of the Myspace artist network is around 7 million, then this is close to the 0.25% sampling ratio suggested for accurate degree distribution estimation in sampled networks. Note that the sampling ratio is not sufficient for estimating other topological metrics such as the clustering coefficient and assortativity [Kwak et al., 2006]; such global measures are not required for this work.

With snowball sampling there is a tendency to over-sample hubs because they have many links and are typically picked up early in the breadth-first sampling. This effect reduces the degree distribution exponent by introducing a higher proportion of nodes with high connectivity than are seen in the complete network, producing a heavier tail but preserving the overall power-law nature of the network [Lee et al., 2006].

[12]Where N is the number of friends, typically 10^3 but in some cases of the order 10^7.

[13]The artist is *Karna Zoo*, Myspace url: http://www.myspace.com/index.cfm?fuseaction=user.viewProfile&friendID=134901208

	n	m	$\langle k \rangle$	l	d_{max}
undirected	15478	91326	11.801	4.479	9
directed	15478	120487	15.569	6.426	16

Table 3.1: The network statistics for the Myspace artist network sample where n is the number of nodes, m is the number of edges, $\langle k \rangle$ is the average degree, l is the mean geodesic distance, and d_{max} is the diameter, as defined in Section 3.2.1.1.

3.3.2 Network Analysis of the Myspace Artist Network Sample

The Myspace artist network sample exhibits many of the network characteristics common to social networks and other real-world networks. Some of the network's statistics are summarized in Table 3.1.

We see that the Myspace artist network is like many other social networks in its "small world" characteristics – having a small diameter and geodesic distance. Additionally, in previous work, it has been shown that the Myspace artist network is assortative with respect to genre labels – that is, artists preferentially form connections with other artists that have the same genre labels [Jacobson and Sandler, 2008].

Although the network is constructed as a directed network, for some of our experiments we convert to an undirected network to simplify analysis. This conversion is done to reduce complexity for analysis and to better examine the reflexive properties that are present in the broader mutual friend connections of the whole Myspace network. Each edge is considered bi-directional, that is $(i, j) \simeq (j, i)$, and if a reflexive pair of edges existed in the directed graph, only one bi-directional edge exists in the undirected graph.

The degree distribution for this undirected reduction network is plotted in Figure 3.2 on a log-log scale. It is common to find a power-law degree distribution in social networks [Newman, 2003]. However, exponential degree distributions have been reported previously in some types of music recommendation networks [Cano et al., 2006]. This is especially true for networks with imposed degree limits. For moderate degree values ($35 < k < 200$), our sample shows a power-law distribution. For lower degree values, the distribution is closer to exponential. This may be related to the fact that our network has an out degree limit imposed by Myspace restricting the maximum number of top friends ($k_{out} \leq 40$). The power-law fit also breaks down for high values of k – most likely due to the limited scope of our sample. Similar "broad-scale" degree distributions have been reported for citation networks and movie actor networks [Amaral et al., 2000]. A more detailed analysis of this Myspace artist

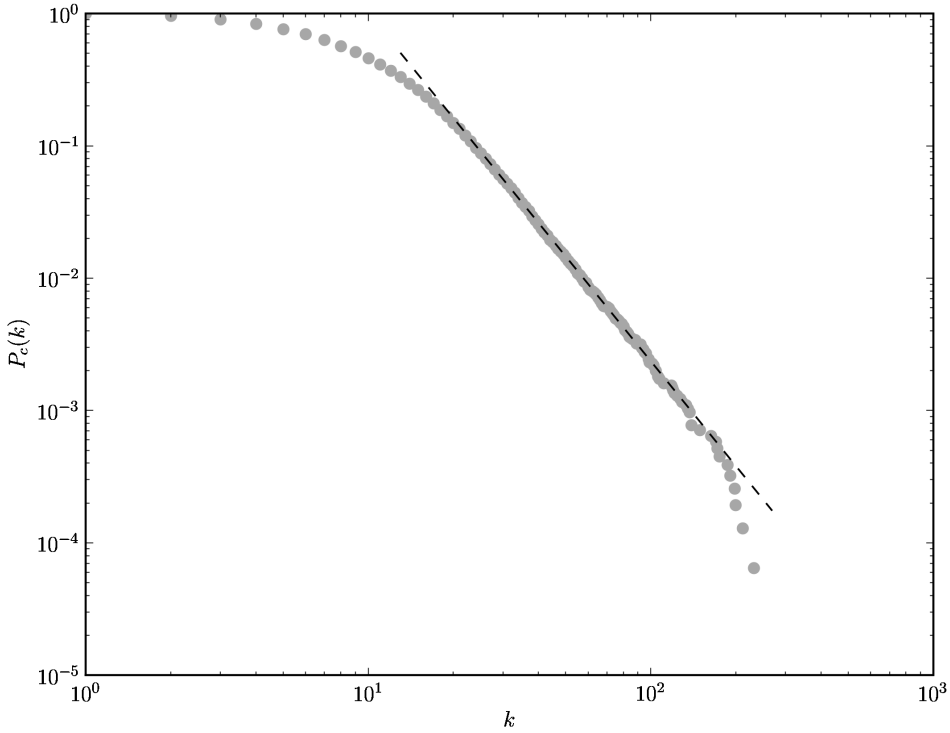


Figure 3.2: The cumulative degree distributions for the Myspace artist network sample. For moderate values of k, the distribution follows a power-law (indicated by the dotted line), but for low and high values the decay is exponential.

network can be found in [Jacobson and Sandler, 2008].

3.3.3 Community Structure

We apply two community detection algorithms to our network sample – the greedy optimization of modularity from Clauset et al. [2004] and the walktrap algorithm as described by Pons and Latapy [2005]. Both of these algorithms are reasonably efficient for networks of our size and both algorithms can be easily adapted to incorporate audio-based similarity measures (see Jacobson et al. [2008] and Section 3.4.3).

3.3.3.1 Greedy Modularity Optimization

Modularity is a network property that measures the appropriateness of a network division with respect to network structure. Modularity can be defined in several different ways [Costa et al., 2007]. In general, the modularity Q captures the relationship between the number of edges within communities and the expected number of such edges. Let $A_{ij} \in 0, 1$ be an element of the network's adjacency matrix and suppose the nodes are divided into communities such that node i belongs to community C_i. We choose the definition of modularity

Q as the fraction of edges within communities minus the expected value of the same quantity for a random network of the same size and degree distribution. Then Q can be calculated as

$$Q = \frac{1}{2m} \sum_{ij} \left(A_{i,j} - \frac{d_i d_j}{2m} \right) \delta_{C_i C_j} \tag{3.2}$$

where the $\delta_{C_i C_j}$ function is 1 if $C_i = C_j$ and 0 otherwise, m is the number of edges in the graph, and d_i is the *degree* of node i – that is, the number of edges incident on node i. The sum of the term $\frac{d_i d_j}{2m}$ over all node pairs in a community represents the expected fraction of edges within that community in an equivalent random network where node degree values are preserved.

If we consider Q to be a benefit function we wish to maximize, we can then use an agglomerative approach to detect communities – starting with a community for each node such that the number of partitions $|\mathcal{P}| = n$ and building communities by amalgamation. The algorithm is greedy, finding the changes in Q that would result from the merge of each pair of communities, choosing the merge that results in the largest increase of Q, and then performing the corresponding community merge. It can be proven that if no community merge will increase Q the algorithm can be stopped because no further modularity optimization is possible [Clauset et al., 2004]. Using efficient data structures based on sparse matrices, this algorithm can be performed in time $\mathcal{O}(m \log n)$.

3.3.3.2 Random Walk: Walktrap

The walktrap algorithm uses random walks on G to identify communities. Because communities are more densely connected, a random walk will tend to be 'trapped' inside a community – hence the name "walktrap".

At each time step in the random walk, the walker is at a node and moves to another node chosen randomly and uniformly from its neighbours. The sequence of visited nodes is a *Markov chain* where the states are the nodes of G. At each step the transition probability from node i to node j is $P_{ij} = \frac{A_{ij}}{d_i}$ which is an element of the transition matrix P for the random walk. We can also write $P = D^{-1}A$ where D is the diagonal matrix of the degrees ($\forall i, D_{ii} = d_i$ and $D_{ij} = 0$ where $i \neq j$).

The random walk process is driven by powers of P: the probability of going from i to j in a random walk of length t is $(P^t)_{ij}$ which we will denote simply as P^t_{ij}. All of the transition probabilities related to node i are contained in the i^{th} row of P^t denoted as $P^t_{i\bullet}$. We then define an inter-node distance measure

for a given value of t:

$$r_{ij} = \sqrt{\sum_{k=1}^{n} \frac{(P_{ik}^t - P_{jk}^t)^2}{d_k}} = \|D^{-\frac{1}{2}} P_{i\bullet}^t - D^{-\frac{1}{2}} P_{j\bullet}^t\| \tag{3.3}$$

where $\|.\|$ is the Euclidean norm of R^n. This distance can also be generalised by averaging to a distance between communities: $r_{C_i C_j}$ or to a distance between a community and a node: $r_{C_{ij}}$.

We then use this distance measure in our algorithm. Again, the algorithm uses an agglomerative approach, beginning with one partition for each node ($|\mathcal{P}| = n$). We first compute the distances for all adjacent communities (or nodes in the first step). At each step k, two communities are chosen based on the minimisation of the mean σ_k of the squared distances between each node and its community:

$$\sigma_k^2 = \frac{1}{n} \sum_{C_i \in \mathcal{P}_k} \sum_{i \in C_i} r_{iC_i}^2 \tag{3.4}$$

Direct calculation of this quantity is known to be NP-hard [Pons and Latapy, 2005], so instead we calculate the variations $\Delta\sigma_k$. Because the algorithm uses a Euclidean distance, we can efficiently approximate these variations as

$$\Delta\sigma(C_1, C_2) = \frac{1}{n} \frac{|C_1||C_2|}{|C_1| + |C_2|} r_{C_1 C_2}^2 \tag{3.5}$$

The community merge that results in the lowest $\Delta\sigma$ is performed. We then update our transition probability matrix

$$P_{(C_1 \cup C_2)\bullet}^t = \frac{|C_1| P_{C_1\bullet}^t + |C_2| P_{C_2\bullet}^t}{|C_1| + |C_2|} \tag{3.6}$$

and repeat the process updating the values of r and $\Delta\sigma$ then performing the next merge. After $n-1$ steps, we get one partition that includes all the nodes of the network $\mathcal{P}_n = \{N\}$. The algorithm creates a sequence of partitions $(\mathcal{P}_k)_{1 \leq k \leq n}$. Finally, we use modularity to select the best partition of the network, calculating $Q_{\mathcal{P}_k}$ for each partition and selecting the partition that maximizes modularity.

Because the value of t is generally low (we use $t = 4$, selected empirically), this community detection algorithm is more scalable then greedy modularity optimization. For most real-world networks, where the graph is sparse, this algorithm runs in time $\mathcal{O}(n^2 \log n)$ [Pons and Latapy, 2005]. Note though, the optimized greedy modularity algorithm scales significantly better for sparse graphs than the walktrap algorithm – $\mathcal{O}(m \log n)$ versus $\mathcal{O}(n^2 \log n)$ – and in

our implementation is faster by an order of magnitude on our sample graph.

3.3.4 Summary

In an effort to create an experimental dataset, the Myspace social network's artist network was sampled. The sample was taken via random entry and a breadth-first walk. Basic analysis of this sample set shows it to conform to norms of other studied social networks. Further an explanation of community structural analysis techniques were laid out, from which to perform multimodal analysis and measurement of the sample. With this understanding of the basic properties of our data set, we can now go forward with experimentation using hybrid distance techniques.

3.4 Hybrid Methods of Distance Analysis

To move towards well-formed uses of both social and acoustic notions of distance, a better understanding of the relationship between these two spaces is required. We therefore conduct a series of experiments to analyse the effect of combining social and content-based distance. Our first two experiments are concerned with distance between pairs of nodes (both artists and songs) in our graph; the third experiment looks into the affect that acoustic distance measurements have in the detection of community structure. These experiments are presented as follows.

1. The geodesic distance between all pairs of artists within the sample are compared to the acoustic similarity of songs associated with each artist.

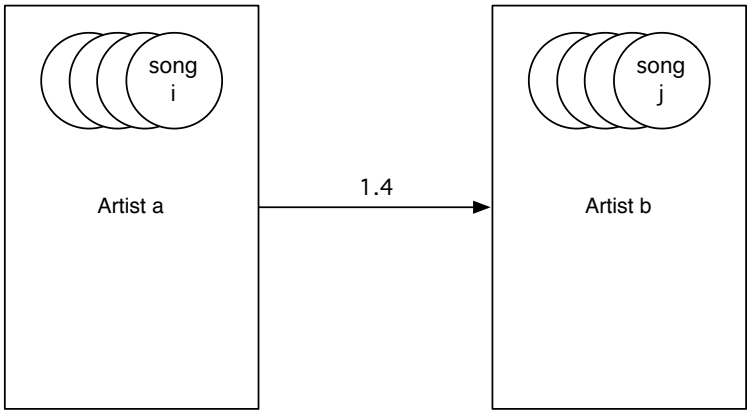
2. Maximum flow analysis is used to analyse the artist social space.

   (a) This measure is compared to the same artist-based acoustic similarity used in item 1.

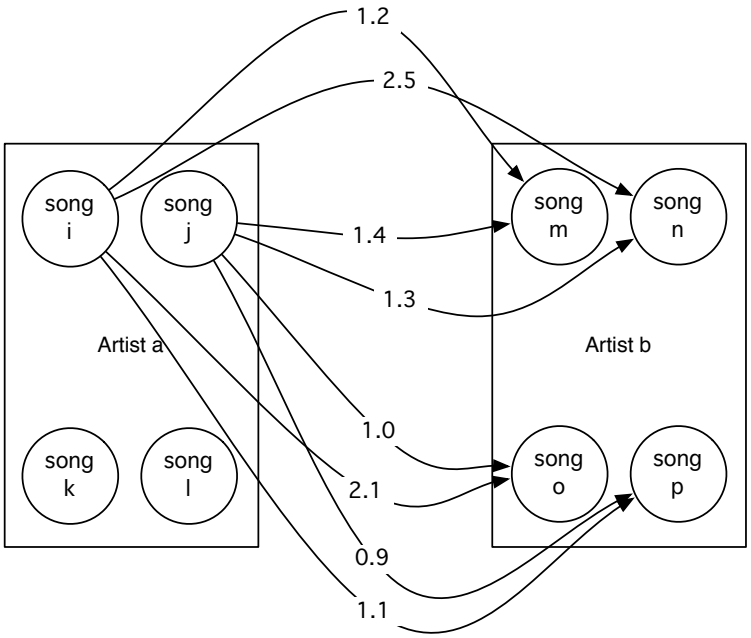
   (b) An additional song-to-song acoustic metric generated by the Marsyas software suite is also used.

3. Community segmentation and structural analysis are explored as a further means of understanding the interaction between these two spaces.

Some of this work requires a network of songs rather than artists (as we sampled in Section 3.3.1). An unweighted graph between songs can be constructed by simply applying the artist connections to their associated songs; weights can be assigned to these song-to-song edges individually, for example based on acoustic dissimilarity between pairs of songs computed with the methods described in this section. These node relationships are illustrated in Figure 3.3.



(a) The sampled artist to artist relationship



(b) The expanded artist relationship, with songs as nodes. Note that the connections of song k and song l have been omitted for clarity.

Figure 3.3: A comparison of sampled and song expanded means of representing the relationship between artists.

MFCCs are extracted from each audio signal using a Hamming window on 8192 sample FFT windows with 4096 sample overlap. All MFCCs are created with the fftExtract tool[14]. For each artist node a GMM is built from the concatenation of MFCC frames for all songs found on each artist's Myspace page. Generally artists have between 1 and 4 songs, although some artists have many more. The mean number of songs is slightly more than 3.5 per artist. An $n \times n$ matrix is populated with the earth mover's distance λ_{ij} between the GMMs corresponding to each pair of artist nodes in the sample. As a second acoustic dissimilarity measure, the software suite Marsyas[15] is used in the exact configuration that was used in the MIREX 2009 Audio Similarity and Retrieval[16] task to generate MFCC-based average value vectors per song and then to generate an $n \times n$ Euclidean distance matrix of these songs. These distance matrices are used to draw λ values to compare against the song expanded graph as detailed above.

3.4.1 Geodesic Paths

The relation between audio signal dissimilarity and the geodesic path length is first examined using a box and whisker plot. The plot is shown in Figure 3.4. These dissimilarities are grouped according to the geodesic distance in the undirected network between the artist nodes i and j, d_{ij}. There appears to be no clear correlation between these λ values and geodesic distance. The Pearson product-moment correlation coefficient confirms this giving a ρ of -0.0016. This should be viewed in the context of the number of pairwise relationships used, implying it is stable, at least for the community of artists found via this sample of the network. Further, it should be noted that our approach to audio-based dissimilarity results in measures which are mostly orthogonal to network structure [Fields et al., 2008a].

3.4.2 Maximum Flow

In our Myspace top friends graph, the maximum flow is measured on the directed and undirected reduction of the unweighted graph from the source artist node to the sink artist node. This extends the work of Fields et al. [2008b] by applying an additional acoustic distance measure (that of the Marsyas entries into MIREX) and examining all the results via means of mutual information.

3.4.2.1 Experiment

The maximum flow value is calculated, using the snowball sample entry point as the fixed source against every other node in turn as a sink, yielding the number

[14] source code at http://omras2.doc.gold.ac.uk/software/fftextract/
[15] http://marsyas.info/
[16] http://music-ir.org/mirex/2009/results/abs/GTfinal.pdf

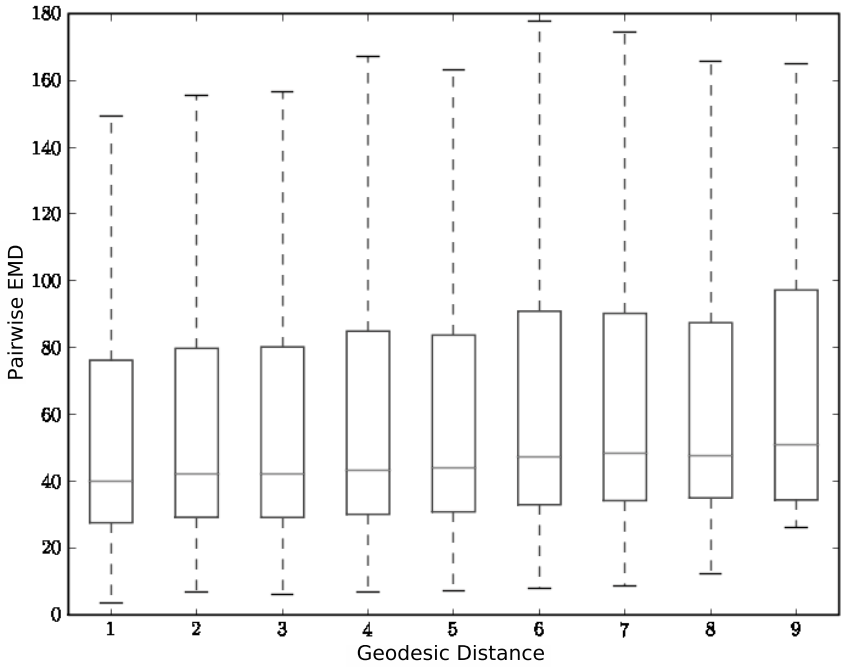


Figure 3.4: The box and whisker plot showing the spread of pair-wise artist dissimilarity grouped by geodesic distance as found on the artist graph. The whiskers cover the second and seventh octiles beyond the inner quartiles covered in each box.

of edges connecting each sink node to the entry point node at the narrowest point of connection. The acoustic distances are then be compared to these maximum flow values.

In order to better understand a result from analysis of our Myspace sample, a baseline for comparison must be used. To that end, we examine random permutations of the node locations. In order to preserve the overall topology present in the network, we perform this randomization by shuffling the artist label and associated music attached to a given node on the network. This is done ten fold, creating a solid baseline to test the null hypothesis that the underlying community structure is not responsible for any correlation between maximum flow values and λ_{ij} from either of the two acoustic dissimilarity measures.

3.4.2.2 Results

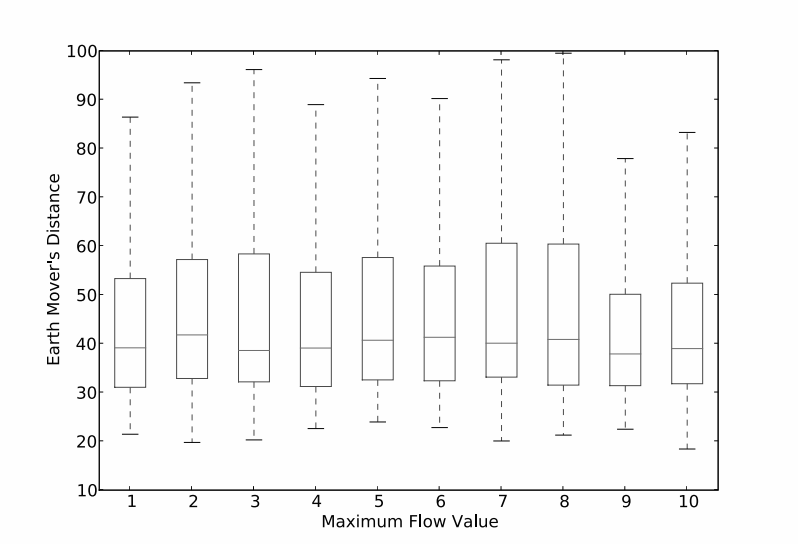
The results of this experiment show no simple relationship between the sampled network and the randomized network. This can be seen in Table 3.2 and in Figures 3.5 and 3.6. There is an increase in the median EMD for the less well-connected (*i.e.* lower maximum-flow value) node pairs in the Myspace sample graph, though this is not significant enough to indicate a correlation, while the randomized permutations are near flat. Perhaps the easiest way to examine the relationship between the sampled graph and randomized one is through the deltas of each group's median from the entire dataset median. This data is shown in the second and fourth column in Table 3.2 and Figure 3.7. Note especially both the EMD GMM acoustic distance and Marsyas generated Euclidean distance have similar performance when viewed in this way. Further, the Kruskal-Wallis one-way ANOVA results for both the sample graph and averaged across the 10 fold permutations are shown in Table 3.3.

Additionally, we calculated the mutual information between the flow network and both of the acoustic distance measures[17]. These can be seen, along with the entropy of each set in Table 3.4. Here we can beyond simply looking at an implied near independence. The mutual information between the maximum flow values and either of two acoustic distance measure is a small fraction of the entropy of either respective set of distances.

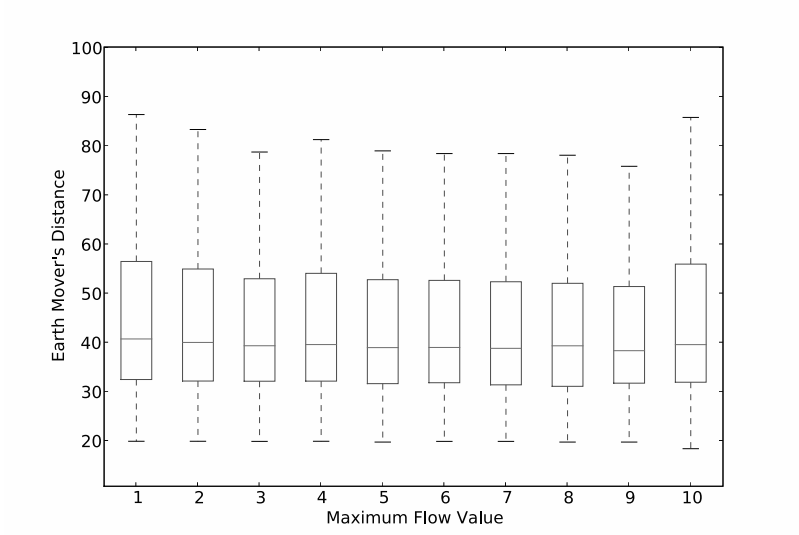
3.4.3 Using Audio in Community Detection

Both community detection algorithms described in Section 3.3.3 are based on the adjacency matrix A of the graph. This allows us to easily extend these algorithms to include audio-based similarity measures. We simply insert an inter-node similarity value for each non-zero entry in A. We calculate these

[17] All mutual information and related entropy calculations in this work are calculated using pyentropy, available at http://code.google.com/p/pyentropy/, a python library for performing information theoretic analysis on data distributions [Ince et al., 2009].

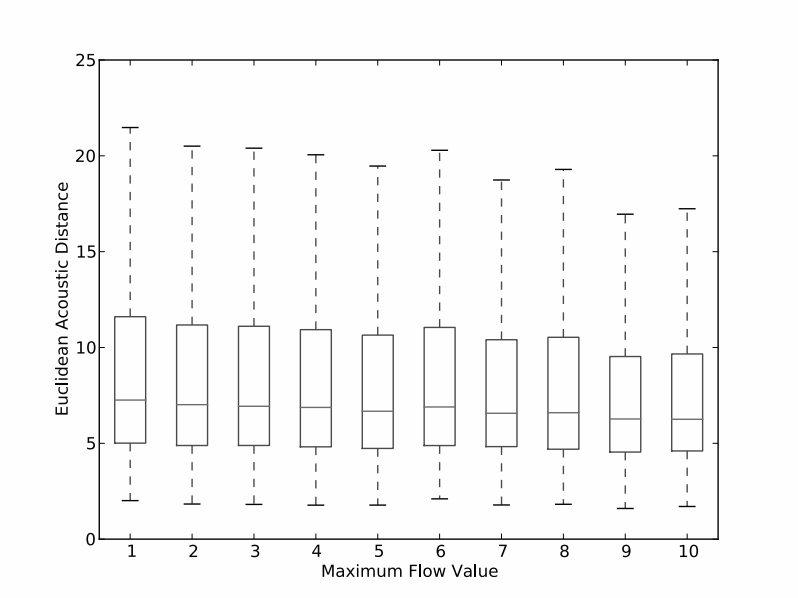


(a) The EMD distribution on the sampled graph

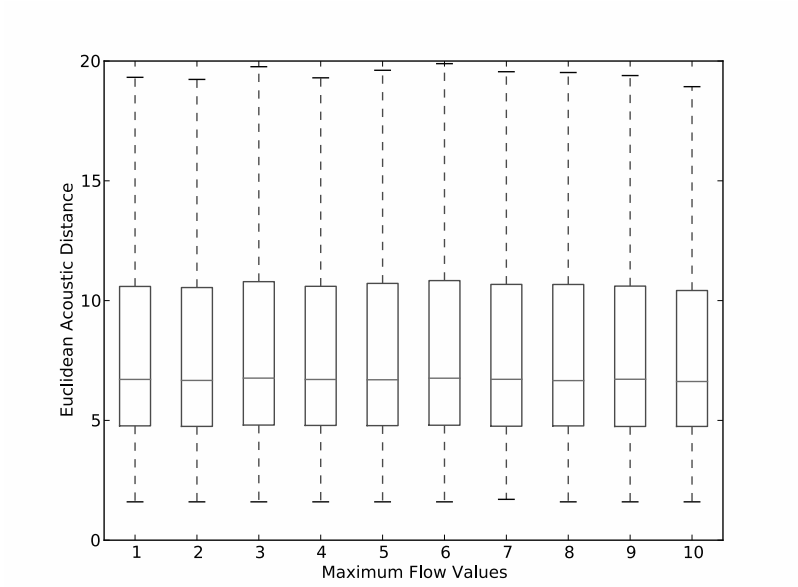


(b) The EMD distribution on the random permutations of the graph, maintaining the original edge structure.

Figure 3.5: The box and whisker plots showing the distribution of EMD grouped by maximum flow value between artists on the Myspace social graph and the randomized permutations of the graph.



(a) The Euclidean distance distribution on the sampled graph



(b) The Euclidean distance distribution on the random permutations of the graph, maintaining the original edge structure.

Figure 3.6: The box and whisker plots showing the distribution of Euclidean distance, grouped by maximum-flow value between artists on the Myspace social graph and the randomized permutations of the graph.

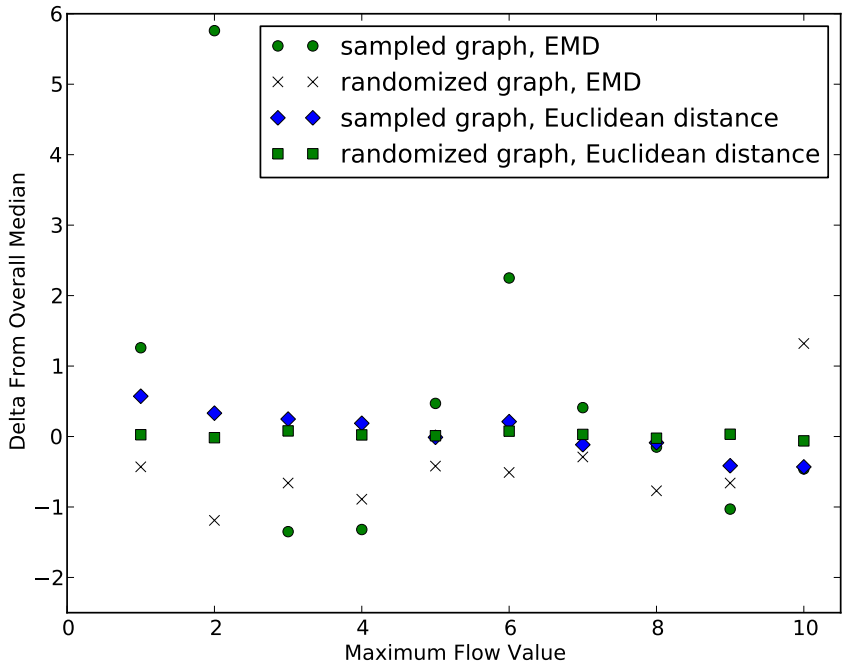


Figure 3.7: The deltas from the global median for each maximum flow value group of acoustic distance values, from the sampled graph and the randomized graph.

similarity values using both the earth-mover's distance and Marsyas' audio-based analysis methods described in Section 3.4. Dissimilarity values from these methods must be converted to similarity values to be applied to the community detection algorithms. We do this by taking the reciprocal of each dissimilarity:

$$A_{ij} = \begin{cases} \lambda_{ij}^{-1} & \text{if nodes } i \text{ and } j \text{ are connected} \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

3.4.3.1 Genre Entropy

Now that we have several methods for detecting community structures in our network, we need a means of evaluating the relevance of these structures in the context of music. Traditionally, music and music artists are classified in terms of *genre*. If the structure of the Myspace artist network is relevant to music, we would expect the communities identified within the network to be correlated with musical genres. That is, communities should contain nodes with a more homogeneous set of genre associations than the network as a whole.

In our sampling of the Myspace network (described in Section 3.3.1 above), we collected genre tags that are associated with each artist. In order to measure

the diversity of each community with respect to genre we use a variant of Shannon entropy we call *genre entropy* S. This approach is similar to that of Lambiotte and Ausloos [2006]. For a given community C_k with the set of $L(C_k)$ genres, drawn from the set L total genres, we calculate genre entropy as:

$$S_{C_k} = -\sum_{\gamma \in L(C_k)} P_{\gamma|L(C_k)} \log P_{\gamma|L(C_k)} \tag{3.8}$$

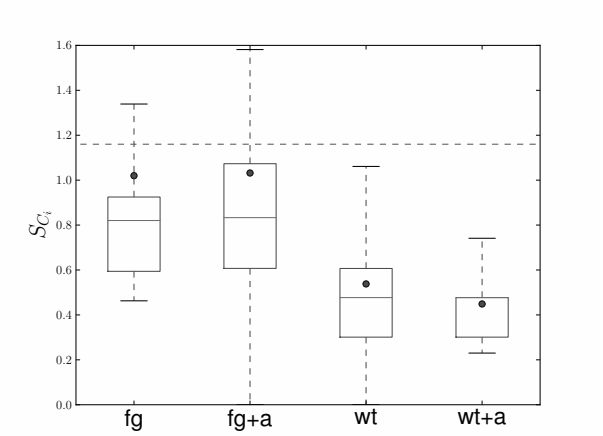
where $P_{\gamma|L(C_k)}$ is the probability of finding genre tag γ in community C_k. As the diversity of genre tags in a community C_k increases, the genre entropy S_{C_k} increases. As the genre tags become more homogeneous, the value of S_{C_k} decreases. If community C_k is described entirely by one genre tag then $S_{C_k} = 0$. We can calculate an overall genre entropy S_G by including the entire network sample. In this way, we can evaluate each community identified by comparing S_{C_k} to S_G. If the community structures in the network are related to musical genre, we would expect the communities to contain more homogeneous mixtures of genre tags. That is, usually, we would expect $S_{C_k} \leq S_G$. However, as community size decreases the genre entropy will tend to decrease because fewer tags are available. To account for this, we create a random partitioning of the graph that results in the same number of communities with the same number of nodes in each community and calculate the corresponding genre entropies S_{rand} to provide a baseline.

If an artist specified no genre tags, this node is ignored and makes no contribution to the genre entropy score. In our data set, 2.6% of artists specified no genre tags.

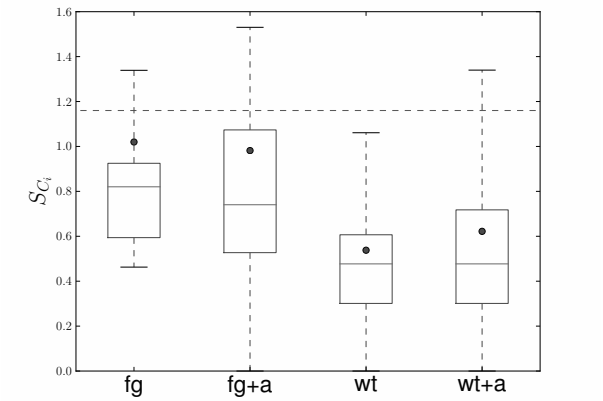
3.4.3.2 Results

The results of the various community detection algorithms are summarized in Figure 3.8 and Table 3.5. When the genre entropies are averaged across all the detected communities, we see that for every community detection method the average genre entropy is lower than S_G as well as lower than the average genre entropy for a random partition of the graph into an equal number of communities. This is strong evidence that the community structure of the network *is* related to musical genre.

It should be noted that even a very simple examination of the genre distributions for the entire network sample suggests a network structure that is closely related to musical genre. Of all the genre associations collected for our data set, 50.3% of the tags were either "Hip-Hop" or "Rap" while 11.4% of tags were "R&B". Smaller informal network samples, independent of our main data set, were also dominated by a handful of similar genre tags (*i.e.* "Alternative",



(a) Audio weights are Earth Mover's Distance



(b) Audio weights are Euclidean distance generated by Marsyas

Figure 3.8: Box and whisker plots showing the spread of community genre entropies for each graph partition method where gm is greedy modularity, gm+a is greedy modularity with audio weights, wt is walktrap, and wt+a is walktrap with audio weights. The horizontal line represents the genre entropy of the entire sample. The circles represent the average value of genre entropy for a random partition of the network into an equivalent number of communities. (a) uses the Earth Mover's Distance for audio weight, (b) uses Euclidean distance from Marsyas.

"Indie", "Punk"). In context, this suggests our sample was essentially "stuck" in a community of Myspace artists associated with these particular genre inclinations. However, it is possible that these genre distributions are indicative of the entire Myspace artist network. Regardless, given that the genre entropy of our entire set is so low to begin with it is an encouraging result that we could efficiently identify communities of artists with even lower genre entropies.

Without audio-based similarity weighting, the fast-greedy modularity algorithm (fg) produces communities with significantly higher genre entropy when compared as a whole distibution to the communities produced via the walktrap algorithm (wt). However the walktrap algorithm results in almost five times as many communities, so we would expect to result in a lower genre entropies because of smaller community size. Also note that as discussed in Section 3.3.3 the optimized fast-greedy modularity algorithm is considerably faster than the walktrap algorithm.

With audio-based similarity weighting, we see mixed results. Applying audio weights to the fast-greedy modularity algorithm (fg+a) actually increased genre entropies but the differences between fg and fg+a genre entropy distributions are not statistically significant. Audio-based weighting applied to the walktrap algorithm (wt+a) results in a statistically significant decrease in genre entropies compared to the un-weighted walktrap algorithm ($p = 4.2 \cdot 10^{-4}$). These relationships hold true when considering audio-based similarity from either of the two measures used throughout the chapter.

3.4.4 Summary

In an effort to better understand and leverage both social connectivity and content-based dissimilarity, we conducted a number of experiments, in two categories: pairwise distance and community segmentation. Our pairwise distance work showed little in terms of a linear correlation, and when examining the mutual information of the two distance distributions it becomes clear that the two encode largely independent spaces, with a very small information content overlap. When looking into community segmentation, we used genre entropy to see if using acoustic distance would improve the quality of segmentation. This addition of the content-based distance made only a slight difference to the segmentation; however, it is clear that the social structure tightly corresponds to the self-applied genre labels.

3.5 Discussion

We have presented an analysis of the community structures found in a sample of the Myspace artist network. We have applied two efficient algorithms to the task of partitioning the Myspace artist network sample into communities and we

have shown how to include audio-based similarity measures in the community detection process. We have evaluated our results in terms of genre entropy – a measure of genre tag distributions – and shown the community structures in the Myspace artist network are related to musical genre. The communities detected have lower entropy over genre labels than a graph with randomly permuted labels.

We compared the social space of the Myspace sample with content-based acoustic space in two ways in Section 3.4. First the geodesic distances of pairs of artists were compared to the acoustic distance between these pairs of artists. Then maximum flow between pairs of artists was compared to both the acoustic distance between the artists and amongst the artists' songs. While not perfectly orthogonal, the artist social graph and the acoustic dissimilarity matrix clearly encode different relational aspects between artists. This can be clearly seen in the small amount of mutual information shared across the sets of distances. The implication is that using both of these spaces in applications driven by similarity measures will result in much higher entropy in the data available to such an application. This suggests that a recommendation or discovery system that can use both domains well has the potential to perform much better than a similar system that relies on only one domain in isolation.

To understand more completely the contributions of the chapter, we revisit the questions posed at the beginning; what have we learned?

Given that this music is published within a relational space, how can we best use all of the available information to discover new music?

Broadly speaking, our work presented two potential ways to combine the disparate domains of social and content-based space. By weighting the social graph with a measure of acoustic distance, various techniques from complex analysis can be applied. Here we focused on pathfinding and community segmentation. Given the breadth of available techniques from complex networks, we cannot yet say if these works are *best*; however, pathfinding is a natural fit to the construction of playlists and previous work has shown playlists to be excellent vehicles for music discovery [Cunningham et al., 2006; Lamere and Eck, 2007].

Can both social metadata and content-based comparisons be exploited to improve discovery of new material?

While this is similar to the previous question, when looked at this way we can be considerably more definitive. When looking at the entirety of our experimentation, especially the mutual information across distributions seen in Section 3.4.2.2, the answer to this question is a clear yes. While a complete

end-user oriented system remains to be developed, this work shows that such a system would be better served drawn from social and acoustic driven notions of distance and similarity.

Can this crowd-sourced tangle of social-networking ties provide insights into the dynamics of popular music?

On this point a clear conclusion from this work is that the expected linear correlation between social and acoustic distance is not present. So does an artist sound like their friends? While perhaps not what one would first guess, it appears the answer is that while an artist may sound like (*i.e.* similar to) their friends, they don't sound significantly dissimilar to artists that are not (*i.e.* artists which have a high social distance are only slightly further away in acoustic terms than those with a low social distance).

Does the structure of a network of artists have any relevance to music-related studies such as music recommendation or musicology?

This work lays out the parts with which an engaging recommender system could be built or musicological study conducted. This compels further study. As the Myspace artist network is of interest to other researchers, we have converted our graph data to a more structured format. We have created a Web service[18] that describes any Myspace page in a machine-readable Semantic Web format. Using FOAF[19] and the Music Ontology[20][Raimond et al., 2007], the service models a Myspace page in RDF and serializes it as XML RDF. This will allow future applications to easily make use of Myspace network data (*e.g.* for music recommendation).

While it is unclear how best to use all the available information from the wide range of artists and musicians, what this work makes clear is that there are advantages to complex multi-domain notions of similarity in music. By using both acoustic and social data recommender systems have more avenues to pursue to present new material to users in a transparent way. Whether either of these spaces can provide insight into the other remains an open question, though our work tend to show the likely predictability of one space from the other is low. In spite or perhaps because of this separation, and given the sheer quantity of data available on the web, it seems inevitable that these domains will be used in tandem in future music recommendation and musicological study.

[18] available at http://dbtune.org/myspace
[19] http://www.foaf-project.org/
[20] http://musicontology.com/

3.6 Engineering Playlist-Based Applications

In an effort to create domain-specific recommender and discovery systems, we outline two ways to apply this work to end-listener applications. The playlist is an ideal means for this [Flexer et al., 2008; Lamere and Eck, 2007] and such applications could then be evaluated using recommender system standard practice [Adomavicius and Tuzhilin, 2005].

3.6.1 The Max Flow Playlist

To build playlists using both acoustic and social-network data, the Earth Mover's Distance is used between each pair of neighbours as weights on the Myspace sample network. Two artists are then selected, a starting artist as the source node and a final artist as the sink node. One or more paths are then found through the graph via the maximum flow value, generating the list and order of artists for the playlist. The song used for each artist is the most popular at the time of the page scrape. In this way playlists are constructed that are both influenced by timbre similarity and bound by social context, regardless of any relationship found between these two spaces found via the work discussed in Section 3.4. Playlists generated using this technique were informally auditioned, and were found to be reasonable on that basis.

There is clearly potential in the idea of the maximum flow playlist. Based on a small and informal listening test, using either audio similarity measure as a weight the results appear to be quite good, at least from a qualitative perspective. The imposed constraint of the social network alleviates to some extent shortcomings of a playlist built purely through the analysis of acoustic similarity by moving more toward the balance between uniformly acoustically-similar works and completely random movement.

3.6.2 Steerable Optimized Self-Organizing Radio

Using the song-centric graph the following system is in development as a means of deployment and testing. This system is designed to play a continuous stream of songs via an Internet radio stream. The playback system begins with an initial seed song and destination song, then constructs a playlist. While this playlist is being broadcast, anyone tuning into the broadcast is able to vote via a web-based application on the next song to serve as the destination. In order to produce a usable output the vote system presents a list of *nominees*, each selected as a representative track from various communities as segregated via means discussed in Section 3.4.3.

Once the current destination song begins to broadcast, the voting for the next cycle ceases. This destination song is considered the seed song for the next cycle and the song with the most votes becomes the new destination, then the

next playlist will be calculated and its members broadcast. This process will continue for the duration of the broadcast. Once this automatic playlist creation system is allowed to run for a sufficient amount of time, a great deal of user data will be recorded. This would include direct preference feedback, voting behavior, average length of time continuously listened and whether listeners (or at least IP addresses) return. This provides a built-in means of human listener evaluation for these playlists.

We explore this fully automatic radio system further in the next chapter.

	Earth Movers Distance				Marsyas generated Euclidean Distance			
Max Flow	median	deviation	randomized	deviation	median	deviation	randomized	deviation
1	40.80	1.26	39.10	−0.43	7.256	0.571	6.710	0.025
2	45.30	5.76	38.34	−1.19	7.016	0.331	6.668	−0.016
3	38.18	−1.35	38.87	−0.66	6.932	0.247	6.764	0.079
4	38.21	−1.32	38.64	−0.89	6.872	0.187	6.707	0.022
5	40.00	0.47	39.11	−0.42	6.673	−0.011	6.695	0.010
6	41.77	2.25	39.02	−0.51	6.896	0.211	6.761	0.076
7	39.94	0.41	39.24	−0.29	6.568	−0.116	6.714	0.029
8	39.38	−0.15	38.76	−0.77	6.597	−0.087	6.660	−0.023
9	38.50	−1.03	38.87	−0.66	6.270	−0.414	6.717	0.032
10	39.07	−0.46	40.85	1.32	6.253	−0.431	6.623	−0.061

Table 3.2: Node pairs of median acoustic distance values grouped by actual minimum cut values and randomized minimum cut values, shown with deviations from the global medians of 39.53 for EMD and 6.6848 for Euclidean distance. EMD weights are on the left and Euclidean distances as generated by Marsyas are on the right.

	H-value	P-value
From sample	12.46	0.19
Random permutations	9.11	0.43

Table 3.3: The Kruskal-Wallis one-way ANOVA test results of EMD against maximum flow for both the sampled graph and its random permutations. The H-values are drawn from a chi-square distribution with 10 degrees of freedom.

audio distance type	$H(X)$	$H(X\|Y)$	$H(Y)$	$I(X;Y)$
Euclidean distance	3.100	3.00	8.65	0.100
GMM/EMD	3.098	2.723	8.65	0.375

Table 3.4: Entropy values for the acoustic distances and maximum flow values. X is the set of audio distance measurements, Y is the set of maximum flow values.

algorithm	c	$\langle S_C \rangle$	$\langle S_{rand} \rangle$	Q
none	1	1.16	-	-
fg	42	0.81	1.06	0.61
fg+a (gmm)	33	0.81	1.06	0.64
fg+a (mars)	35	0.74	1.06	0.63
wt	195	0.42	1.08	0.61
wt+a (gmm)	271	0.42	1.06	0.62
wt+a (mars)	269	0.42	1.06	0.62

Table 3.5: Results of the community detection algorithms where c is the number of communities detected, $\langle S_C \rangle$ is the average genre entropy for all communities, $\langle S_{rand} \rangle$ is the average genre entropy for a random partition of the network into an equal number of communities, and Q is the modularity for the given partition as defined in Eq. 3.2.

Chapter 4

Steerable Optimizing Self-Organized Radio

> "The anonymous programmers who write the algorithms that control the series of songs in these streaming services may end up having a huge effect on the way that people think of musical narrative – what follows what, and who sounds best with whom. Sometimes we will be the d.j.s, and sometimes the machines will be, and we may be surprised by which we prefer"
> –Sasha Frere-Jones, *You, the D.J. – Online music moves to the cloud*, The New Yorker, 14 June 2010

We detail a fully-automatic, interactive radio system, designed to put the lessons of the analysis presented in Chapter 3 into practice, within an interactive, user-centric, group-playback application. We first expand on and specify ways to improve playlist generation, looking at ways to elicit better queries from a user and consider the implications of novelty curves and expectation. From there the practicalities of the Web as a development platform are discussed, especially as it relates to our application. A detailed view of our interactive model is presented, followed by a complete specification of the deployed system. The chapter concludes with a discussion of evaluation, both those performed and proposed extensions. The system this chapter describes is live and publicly accessible via http://radio.benfields.net. All source code is published under an open source license and available[1].

4.1 Generating Better Playlists

Before discussing the particulars of our system, we will first formalise approaches to generate playlists that better meet the needs and requirements of the listener. In this way we will develop a user-centric specification for any

[1] source available at https://github.com/gearmonkey/sosoradio

playlist-generation system, where the algorithm design and parameterisation is informed by an understanding of the requirements of listeners. For this discussion we will consider playlist generators as information retrieval systems. From this framing, input given to a playlist generator is considered a *query*; similarly, generated playlists are considered *retrieved results*, and we consider their *relevance* to the query.

4.1.1 More Specific Queries

To provide playlists which are more likely to satisfy listeners, systems must be designed to elicit queries that are as specific as practical from users with which to build playlists. As discussed in Chapter 2.8, the majority of existing systems provide an interface to the playlist generator that does not elicit a sufficient amount of specificity.

The most common form this deficiency takes is playlist generation by seed song [Aucouturier and Pachet, 2002; Knees et al., 2006; Logan, 2002]. Here a single song is the only means used to specify the desired playlist. The particulars of each of these approaches is discussed in more detail in Section 2.8.2, but the lack of specificity that effects all of them can be considered as follows. A playlist is a sequence of songs. If those songs, along with others, are projected into a space, a playlist in that space can be considered as an approximation of a *vector* through that space. A single point (song) is not enough information to specify any arbitrary vector. When a playlist generation system only uses single song queries with no further input (implicit or explicit), the vector through space has no specified direction, so a playlist of songs within a radius or heading in a random direction is the only method that can be used, though it may not be what would best meet the user's needs.

As the qualitative results show in Flexer et al. [2008], playlist results are vastly improved by the additional information provided by the a second song being included in a query system. Although this is not the only method of further query specification acknowledged in literature (see Section 2.8.2), the critical point is that any viable novel system must have a means for the user to specify not only a starting position but the trajectory as well.

4.1.2 Novelty Curves and Expectation

Presenting any ordered list of stimuli to a user results in changing perceptions of novelty and expectation as the ordered list is presented. In order to make playlists that better meet a users needs, if relying on content-based features, this information of a novelty curve must be encoded somewhere else. A statistical model-based system similar to that described in Platt et al. [2002] could be extended to work with extracted audio features rather than textual features.

Another approach is to encode the playlists from Ellingham [2007] and Lynskey [2008] and employ these as training data for a heuristic system. The goal of this approach is to apply the dynamic shape of aggregate expert playlists onto new material, though is is unclear whether it is effective without a great deal of human intervention.

When using a dataset with an underlying graph that follows a power-law distribution of linking, novelty can be met through a selection of items that are spread across the degree distribution. Novelty and interest can also be fostered by traversing multiple communities (areas of high linking) in a single playlist. This novelty arises because in order to go from one community to another via a connected path, a constructed playlist must cross nodes with lower linking serving the connection between communities. This leads to a forced variance in degree connectivity, a proxy for novelty, across a playlist.

4.2 The Web as a Platform

Developing software traditionally entails choosing an operating system to target for deployment. As software development tools have grown and matured, it has become easier to create applications that can run on multiple operating systems (OS). There exist a variety of ways to accomplish cross-platform targeting including avoiding OS-specific libraries (Cocoa in Mac OS X for example), developing for a common virtual machine (Java and the Java Virtual Machine or JVM) or the use of interpreted rather than compiled languages (e.g. Python, Ruby or Max/MSP).

The SoSoRadio system as detailed in this chapter requires a persistent internet connection for a user to stream music as it is selected by the system for playback and also to submit requests back to the service as will be discussed in Section 4.3. Given this central requirement, the system is developed using the Web as a platform and by doing so achieve operating system independence by shifting the burden of interoperability to the Web Browser [Taivalsaari et al., 2008]. This approach is becoming more widespread, covering everything from social networking environments such as Facebook[2] to the more traditional business applications available through Google Docs[3]. This treatment, coupled with the use of a collection of standards[4] allows for applications that conform to users expectations and makes for an effective application environment [Fielding,

[2] http://facebook.com

[3] http://docs.google.com

[4] The Web as it is commonly understood is the sum of many commonly agreed upon (or standardized) ways of viewing and transmitting data along with the data and services that meet these standards. These standards are overseen by the World Wide Web Consortium (W3C)(http://www.w3.org/). The core standards that make up the web include the HyperText Markup Language (HTML), Cascading Style Sheets (CSS), and Javascript.

2000].

4.3 Interactivity Model

The inspiration for the interactive model employed in SoSoRadio comes from terrestrial radio music programs; in particular, *request* shows. In these shows, listeners would contact the show while it was on the air with a song they wanted to be included in the upcoming programming (commonly by voice call, historically by handwritten letter, however more recently the contact methods have broadened to include email and text message). While some of these shows would have the entire make-up of a program dictated by this listener request process, the most common format entails a mix of requested songs from the listeners and music selected by the presenter. In the ideal, this non-requested material is selected and ordered to construct coherent transitions from one user request to another. Thus the song selection of the DJ (or music director or whoever is ultimately programming the radio show) is being steered by audience requests while at the same time acting as a curator of those requests through ordering and selection of other material.

4.3.1 Input via Periodic Request

A request given the context of what is currently being listened to provides a system with a more precise input mechanism than the context-free single-seed-song playlists of Aucouturier and Pachet [2002]; Knees et al. [2006]; Logan [2002]. While listening to the system's content, a listener is given the ability to request a song. These two things together, the context of currently – or about to be – played music and the request song, form the playlist query coming from the listener. In this way a more specific query is presented to the playlist-generating system, analogous to start-and-end-songs-based systems as described in Section 2.8.2 such as Flexer et al. [2008].

Taking a cue from request radio shows, the input mechanism for playlist generation is constructed in such a way to allow for variable levels of interaction from the audience. This variability occurs in two directions, from the system's perspective and from the perspective of individual listeners. The system provides for a variable level of request interactivity by dictating how often a request is played and how often content is selected to bridge these gaps. This provides an upper bound on the frequency of request interactivity. The lower bound of request interactivity comes from the individual listeners. A listener may decline to provide a request for any given request period. In this case that listener's request does not factor into the aggregation process. In the event that no listener provides a request for a given period, one is selected at random from the nominee list as described in the following Section.

The request side of the interactive model can be seen in Figure 4.1.

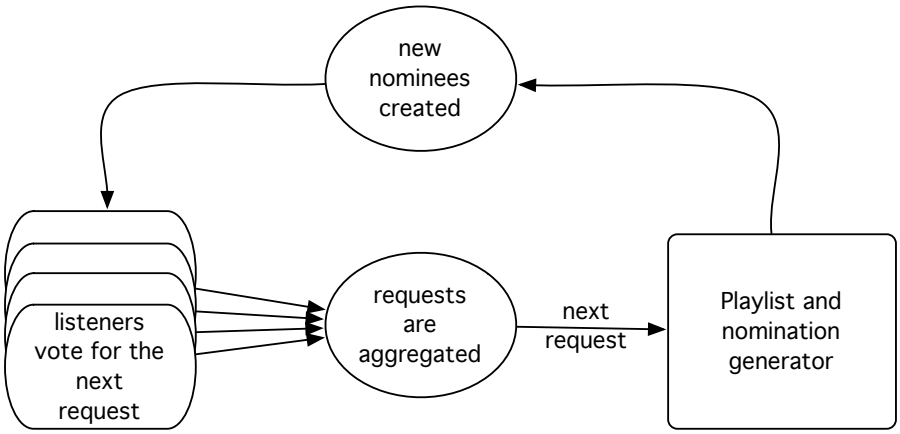


Figure 4.1: The portion of SoSoRadio's interactivity model for generating nominees and eliciting requests

4.3.2 Narrowing Choice

Choice overload is defined in psychological literature as the phenomenon of the difficulty in evaluating and selecting from a number of object increasing exponentially as the number of objects increases. This effect has been shown to be observable in recommender systems, classically by presenting many undifferentiated objects and more recently by showing many (more than 50) objects with a high rating as recommendations to the user [Bollen et al., 2010]. To mitigate the negative effects of choice overload, the SoSoRadio system presents a selection of *nominees*, a subset of tracks from which a listener can select a request track for the period. These nominees are automatically selected to maximise spread across the artist social communities within the collection of tracks available to the system. Further they should be representative of the communities from which they are drawn, as each nominee stands in for all the other tracks in a community.

4.3.3 Eliciting Feedback

Besides listener interaction via periodic requests from nominated tracks, an additional means of feedback is available within the system's interactive model. This feedback is the ability to rate a currently-playing track from one to five, with five being best, while the track is playing. While rating currently-playing music has been used in many existing music recommendation systems, SoSoRadio differs in its interpretation of what exactly is being rated. While the listener is asked to give a rating for the current song playing, this rating is applied to the bi-gram of the currently playing song and the song that was played directly before it; equivalently this can be considered as rating the *edge* connecting two

nodes in a graph, where the nodes are tracks as is the case in the graph described in Section 3.4. This is done to ensure that the context this rating occurs in is not lost. By recording the ratings in this manner, traditional song ratings are available, but so too are more subtle (though sparse) contextual rating uses. For example, this contextual data would make it is possible to examine how a song effects the subsequent expected value of rating from a given user (using emphe.g. value-added analysis Bryk and Weisberg [1976]).

4.4 The System

The SoSoRado system combines the hybrid similarity space described in Chapter 3 with the interactivity model detailed in the previous section, creating a complete automatic music delivery system that responds to its users while curating the underlying musical data in a way that users find compelling.

4.4.1 Overview

The system is composed of a number of disparate processes, communicating via a number of means. Its prototypical deployment spreads these processes over four physical computers, not including the client-side processes. The client-side processes can run on any machine with access to the Web and a modern browser. An overview of the complete system is shown in Figure 4.2.

4.4.2 User Interface

When arriving at the website, a user is presented with a screen like the one seen in Figure 4.3. As the content is dynamically generated based on the currently playing song and nominees, the particular images will vary, but the layout remains the same. In this initial landing page the user is presented with three distinct actionable items: the user can open the stream and listen to the audio content; view the current playlist, see the current track and rate it; and examine the nominees for the current request cycle, including external links for further information, and vote for a nominee.

The left half of the interface shows the current playlist with earlier songs at the top of the list and later songs further toward the bottom of the screen. Within this playlist display, the currently playing song is shown larger than the other songs in the playlist along with an artist picture. There is a drop down selector allowing for the rating of the current song from 1 to 5. Figure 4.4 shows an example screen capture after a rating (in this case a rating of 5 is given) has been indicated by the user. After successfully recording a rating the server sends back the rating, which gets displayed to the user underneath the rating selector, as a means of verification.

The right portion of the screen displays the nominated tracks for the cur-

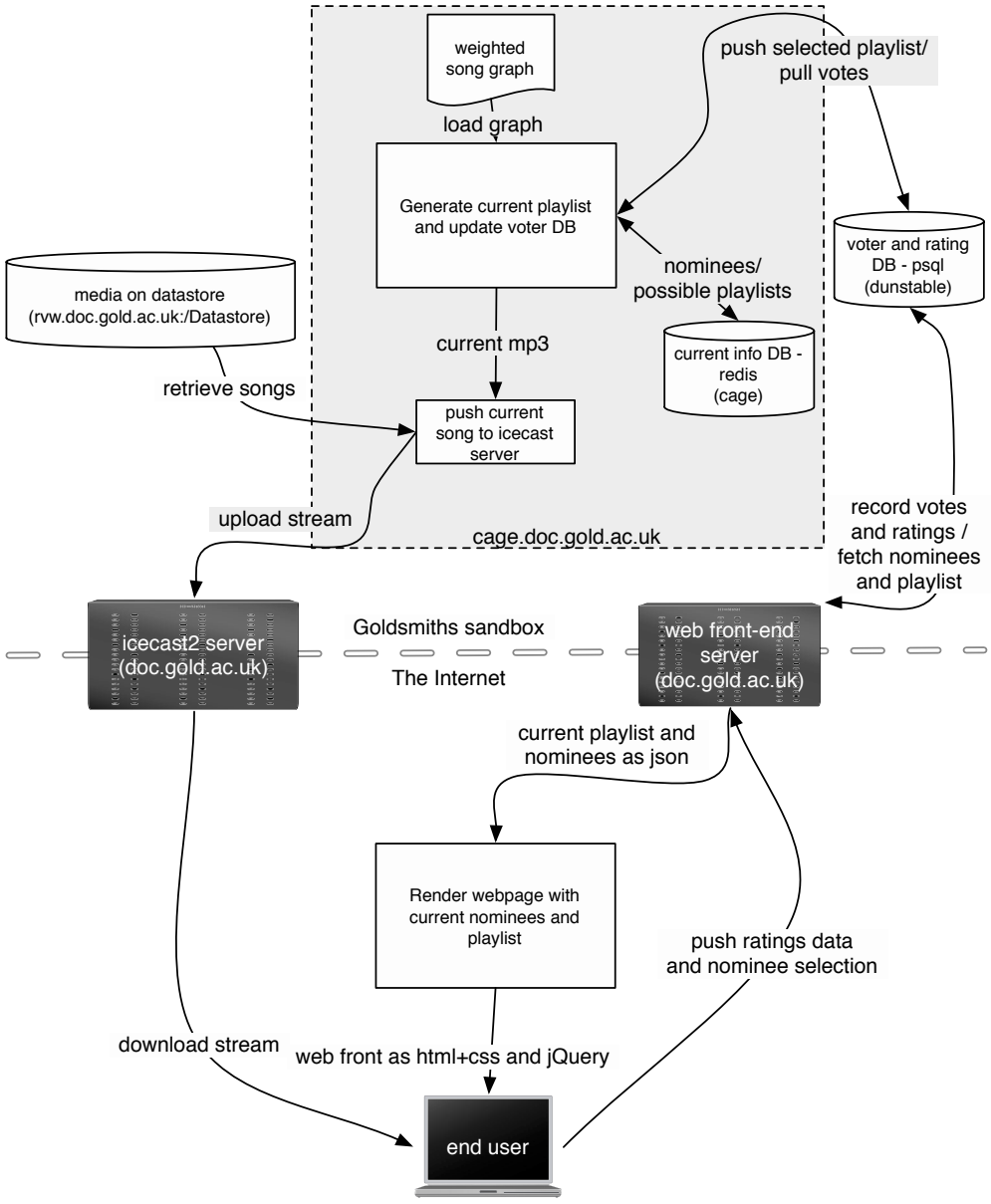


Figure 4.2: A system-wide block diagram of SoSoRadio

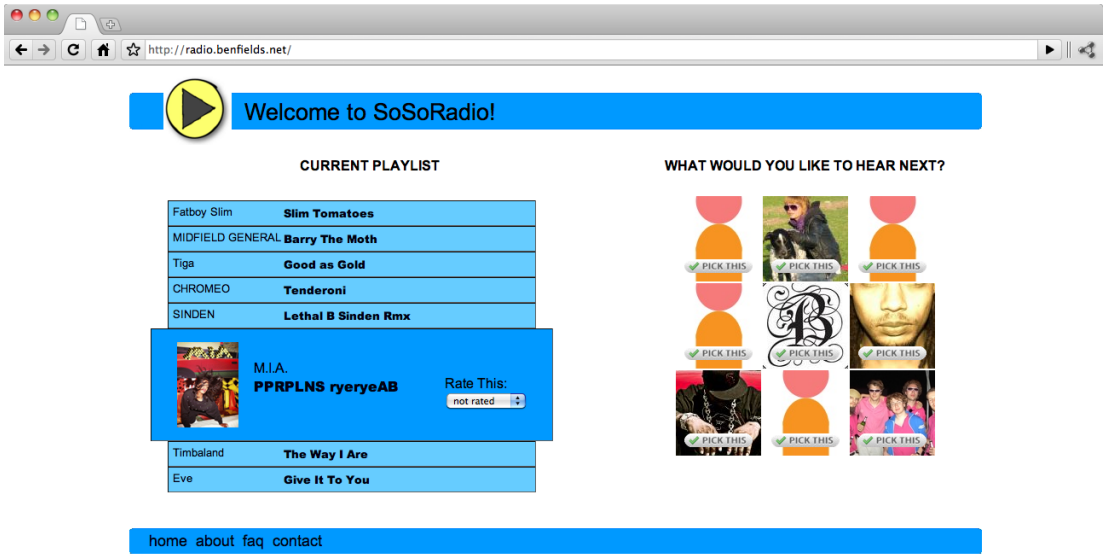


Figure 4.3: Initial landing page for radio user page

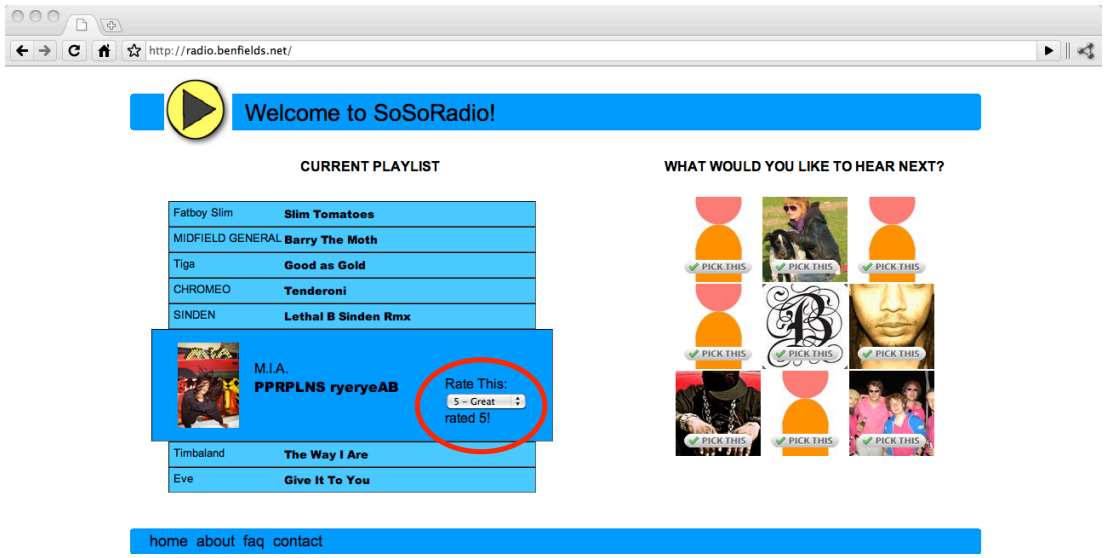


Figure 4.4: After the user has rated the current song

Figure 4.5: Screenshot showing the hover behavior of of the nominees

rent request period and related data including artist and song name, along with an out link to the source social network. Each of the nominated tracks is represented by an icon, automatically created using the center 100×100 pixel square from the artist's profile picture (gathered from the source social network). By default only this icon is seen for each of the nominated tracks, as is seen in the initial screen capture shown in Figure 4.3. When the cursor *hovers*, or is brought over the icon without clicking, the icon is grayed out with the track's text metadata shown (linked to the source social network page) along with a button to vote for the nominated track. This can be seen in Figure 4.5.

Once a user has selected a nominated track the display changes. When the user is hovering the cursor over any of the nominated track icons, the interface appears as it did before a track was selected. However, when the cursor is hovering elsewhere on the screen, the nominee tracks become grayed out, with the current percentile of the vote each nominee has received overlaid on the nominee's icon. An example can be seen in Figure 4.6

In this state, users can listen to the remainder of the current session and also follow to see how their selected nominee is fairing in the voting process. If users want to change their vote they can do so at any time until just before the end of the session when the voting is stopped to generate the next session's playlist.

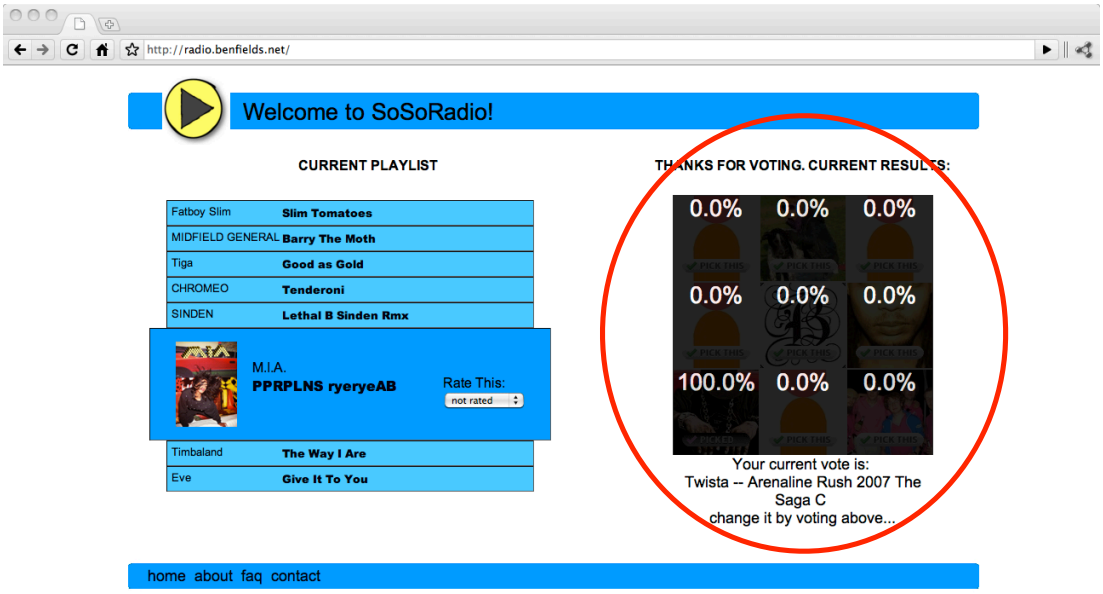


Figure 4.6: Once a user has voted for a nominee the nominees will be greyed, showing the current percentiles of votes for each nominee

4.4.3 Core System

The core system can be considered in three parts: the streaming service itself, the playlist generator and the nomination generator. Each of these systems depends on a weighted-directed-graph representation of the songs available to the system. The prototype live instance of the system uses the songs-as-nodes graph described in Section 3.4, with the weights set as the song-to-song similarity found using Marsyas as described in the same section.

The streaming service is built using icecast libshout[5] via the Python wrapper shout-python. Icecast is an open source streaming audio toolkit that provides stream compatibility with the shoutcast[6] software (available under a commercial license) that predates it.

The playlist generator finds the shortest path through nodes that have not been visited in the past *session* between the end of the current playlist and the next request track. A session is a look back window containing everything that has been played in a fixed previous length of time. In the live prototype, this is 3 hours. The playlist generator also performs the vote aggregation to go find the request song. In the existing system, this aggregation is a simple majority tally mechanism. Once the next request track is determined, the shortest path

[5] http://www.icecast.org/
[6] http://www.shoutcast.com/

is found between the two song as detailed in Section 3.2. Once the playlist is calculated it is pushed to the common database to allow the streaming service to play each track in order and the interface layer for display.

Finally, the nomination generator determines the songs which will be made available to listeners to request for the upcoming request period. The nomination process first breaks the graph of songs into communities as described in Section 3.3.3, using the audio weights applied to edges. For each community, excluding the community containing the last song of the current playlist, a representative song is found by the following method:

1. The duration of the potential playlist formed between the last song of the current playlist and a potential nominee must fall within the period range, set a priori. In the live prototype this is 30 ± 5 minutes.

2. The pagerank [Langville et al., 2008] of each song in the community is then calculated taking the community as a discrete subgraph for the purpose of the pagerank calculation. In this way pagerank serves to describe the inter-community relevance of each track.

3. A song from the top 85^{th} to 95^{th} percentile is selected as the representative nominee track for the community.

At the end of this process each community will have selected a single representative track, with the exception of communities where no tracks can be found that meet the first requirement. From this pool a random subset of 9 is selected and recorded into the database for the interface to access. Note that the size of the subset is arbitrary and can be tuned to optimize the interactive model.

4.5 Playlist Analysis and Evaluation

When evaluating playlists it is important to consider what evaluation is for. In the SoSoRadio system, we seek playlists that balance between popular and unknown material, as a stand in for novelty-curve analysis, and playlists should be enjoyable, by way of generating positive ratings. The SoSoRadio system has been running since May 4^th 2010 and the following analysis is based on a portion of the playlists that have been created from then until November 2010. Some basic statistics for the playlists being used for evaluation can be seen in Table 4.1.

4.5.1 Genre Labels

Myspace artist pages are self-labelled with between zero and three genre labels from a list of options provided by Myspace. These genre labels are used to assess the variance in style seen in the playlists produced by SoSoRadio. Figure 4.7

Number of playlists	857
Avg. number of songs	6.18
Most prevalent genre label	'hip-hop'

Table 4.1: Basic statistics for the evaluation playlists from SoSoRadio

shows a histogram of the number of genre labels used to describe each playlist's artists. While the most common number of labels used to describe all the artists in a playlist is three (the maximum number of labels used to describe a single artist), the majority of playlists use at most six genre labels showing a genre coherence.

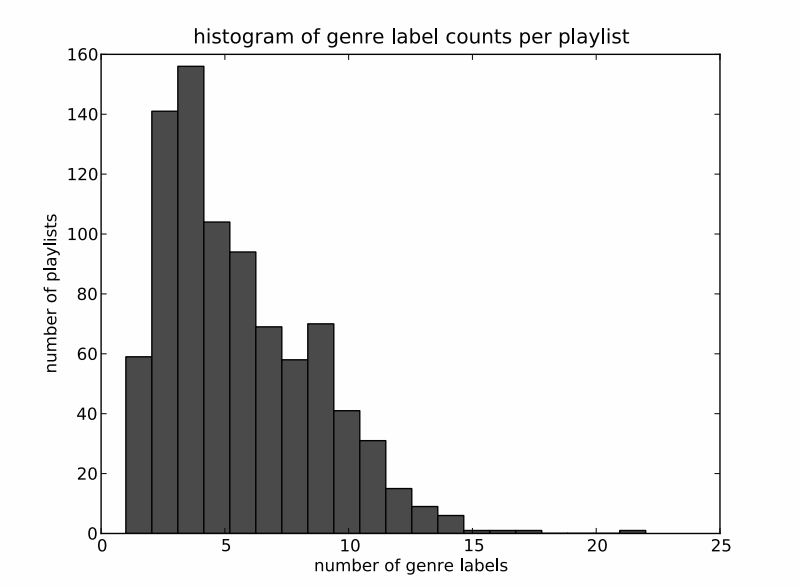


Figure 4.7: Histogram of the number of genre labels occurring in SoSoRadio produced playlists

In addition to considering the coherence of the entire playlist, it is also important to consider the similarity of neighbouring songs in a playlist. While it may be desirable to have some variety in a playlist, this must balance with small changes between neighbouring songs in the majority of cases. This can be measured by finding the change in genre labels from one song's artist to another that neighbours it in playback order, extending the measure used in Knees et al. [2006] to consider multiple genre labels per song. We call this measure *smoothness*. Given two songs i and j, with genre label sets I and J,

the smoothness between them, S_{ij}, is defined as

$$S_{ij} = \frac{|I \cap J|}{\max(|I|, |J|)} \tag{4.1}$$

$|I|$ and $|J|$ are the number of genre labels used to describe songs i and j respectively. For the Myspace artist pages used by the prototypical instance of SoSoRadio, $|I \cap J|$, $|I|$, and $|J|$ are all between zero and three inclusive. This measure is quite straightforward in the case where both songs are created by artists with the same number of genre labels (*e.g.* if two artists are each described by three genre labels and two are the same, the smoothness would be 0.66). However, the case of differing numbers of labels used to describe neighbouring artists is non-obvious enough to warrant a full example. This example is visualised in Figure 4.8.

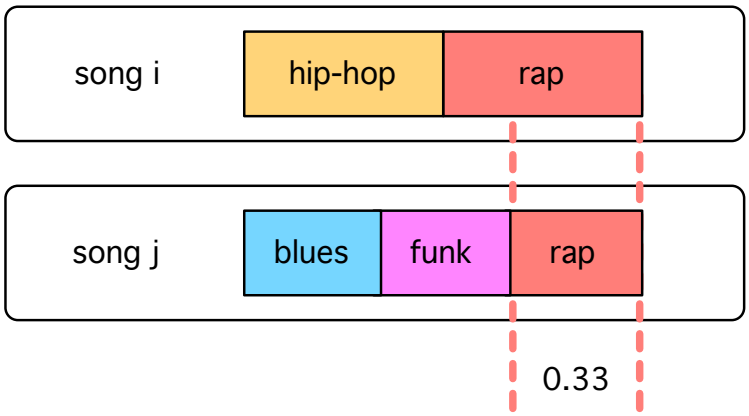


Figure 4.8: An example demonstrating smoothness calculation

Here song i is associated with an artist labelled with the two genre set ('hip-hop', 'rap'). Each of these genre labels can be considered as 0.5 of the full genre description of the artist. Similarly, song j is associated with an artist labeled with three genre set ('blues', 'funk', 'rap'). Each of these genre labels can be considered a 0.33 of the full genre description of artist. Therefore, an examination of the portion of these two genre descriptions shows an overlap of a 0.33. This is equivalent to smoothness found by Equation 4.1. Note that the ratio of overlays in Eq. 4.1 is based on variable genre labels that all carry equal weight, as is the case in Myspace. There are weighted textual descriptions of songs that could be used in a similar fashion (*e.g.* social tags from Last.fm) though richer label sets bring different problems (Section 5.3).

Taking the simple mean of the smoothness of all transitions in a playlist provides a measure of how much stylistic change occurs between any two songs

across a playlist. The histogram of this average for the test set of playlists is seen in Figure 4.9.

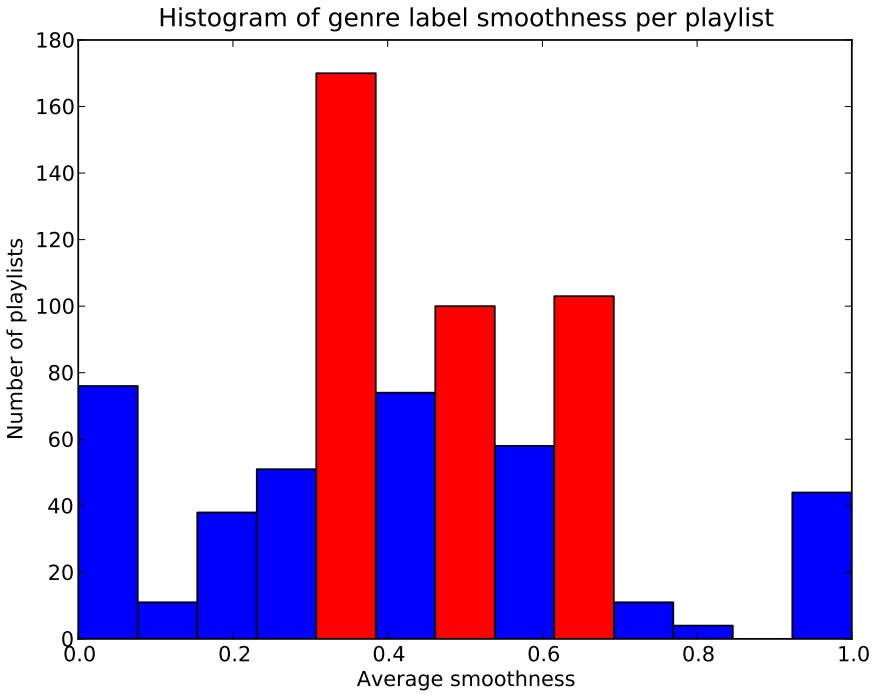


Figure 4.9: The histogram of average smoothness for SoSoRadio playlists. The histogram bins containing 0.33, 0.5 and 0.66 are coloured red. These three values are the smoothness for genre descriptor set overlaps of 1 in 3, 1 in 2, and 2 in 3.

The minimum and maximum smoothness for each playlist is also shown in histograms (Figure 4.10 and Figure 4.11). These histograms give the distribution of the least and most abrupt stylistic change in each of session.

Taken together, the total genre label counts and mean, minimum, and maximum smoothness show SoSoRadio's output to be stylistically heterogeneous while generally keeping neighbouring songs similar, in so far as the genre labels provide an adequate approximation for stylistic description.

4.5.2 Familiarity

The playlists produced by SoSoRadio are also examined on the basis of artist familiarity. *Pageviews*, or the number of times an artist's profile page has been accessed, are used as a measurement of the familiarity as it is reasonable to assume that an artist whose page has been accessed frequently is more well known than an artist whose page has a low pageview count. The examination of pageviews will follow the course of the previous analysis of genre labels, first

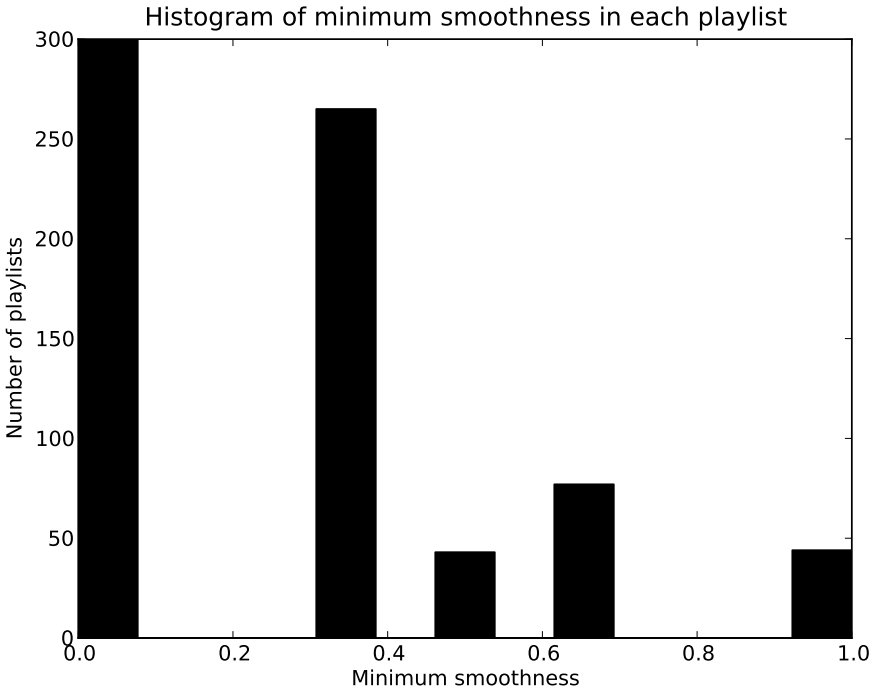


Figure 4.10: Histogram of minimum smoothness in each playlist

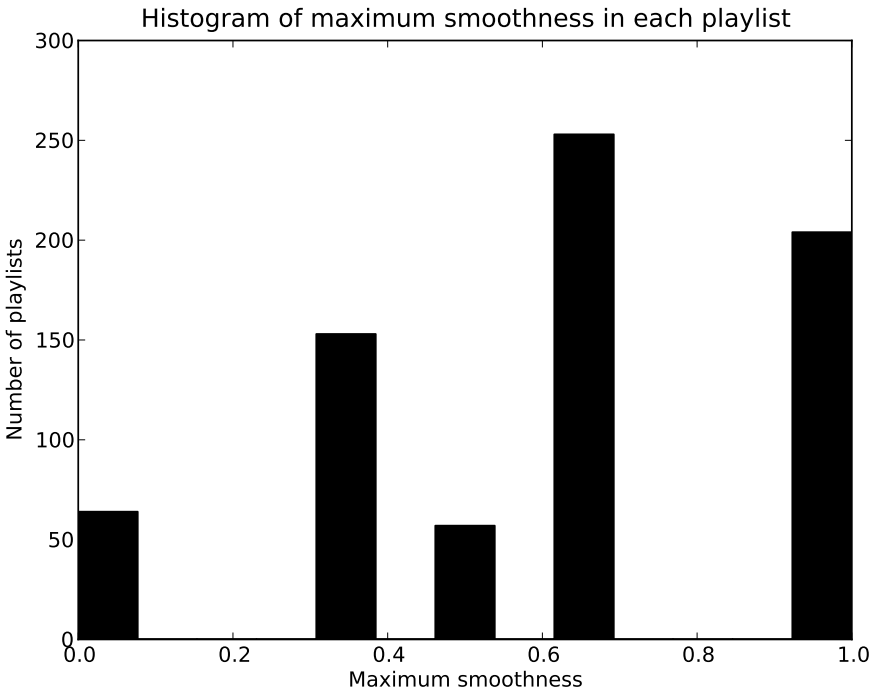


Figure 4.11: Histogram of maximum smoothness in each playlist

looking at the set of songs in each playlist, then examining the transitions.

Figure 4.12 shows the distribution of mean pageviews for the playlists produced by the radio system.

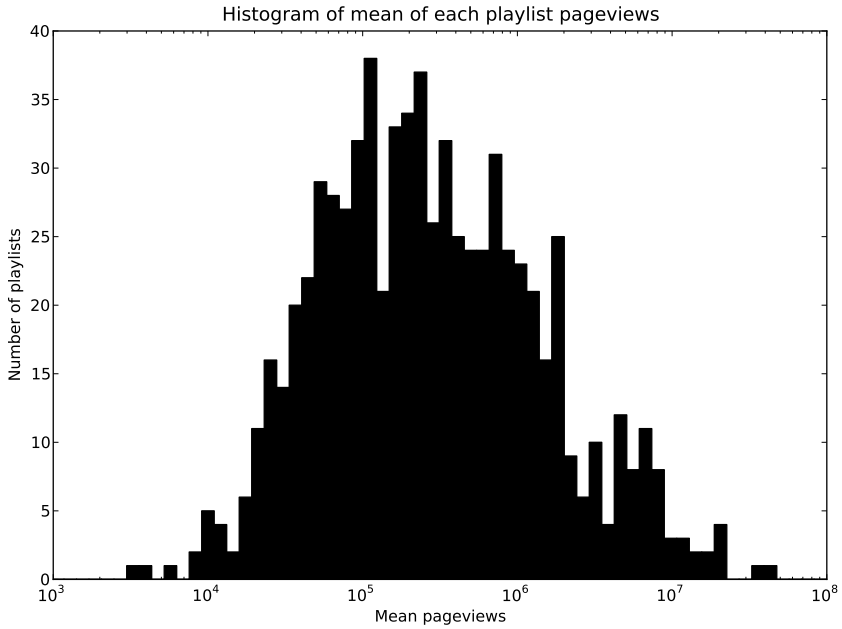


Figure 4.12: A histogram of the mean of each playlist's artists pageviews

Figure 4.13 and Figure 4.14 show the minimum and maximum artist pageviews, for each playlist in the test set. Figure 4.15 superimposes the three proceeding histograms showing the difference in shape between the distributions.

Figure 4.16 show the standard deviation of the same pageviews. The majority of mean pageviews is around 100,000 with bins decreasing in membership as the mean goes either higher or lower. The majority of the variance is lower, approximately 10,000, the smallest bin of the histogram.

Again, an analysis is run on the neighbours, now concerning familiarity. This is investigated by taking the absolute value of the difference between pageviews for each neighbouring song's artist's in a playlist. The mean can then be taken for these values across the entire list. Figure 4.17 shows the histogram of these means.

The minimum and maximum values for the delta pageviews are shown in Figure 4.18 and Figure 4.19. The mean, minimum and maximum delta pageview histogram bin counts are shown together in Figure 4.20.

These analyses show a familiarity that varies in a way that mirrors our un-

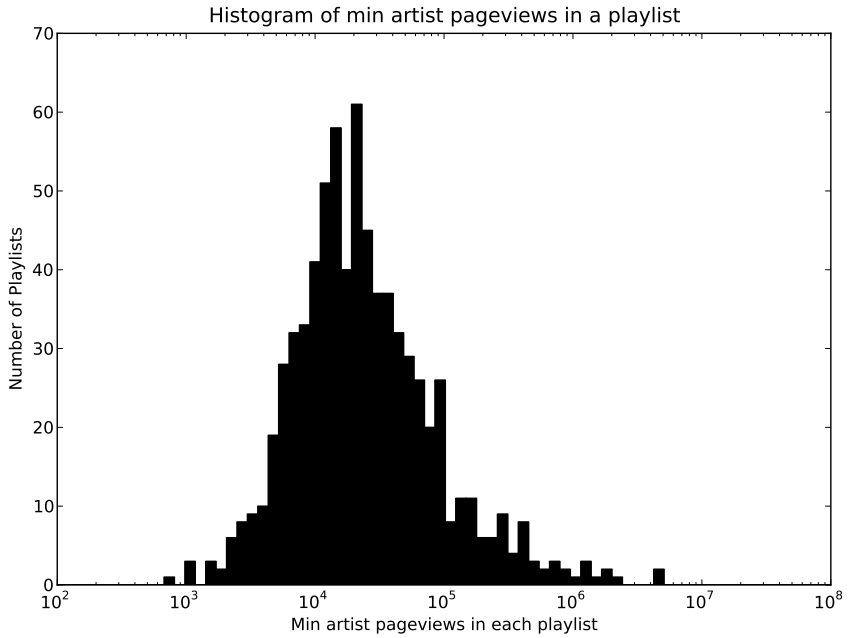


Figure 4.13: Histogram of minimum artist pageviews in a playlist

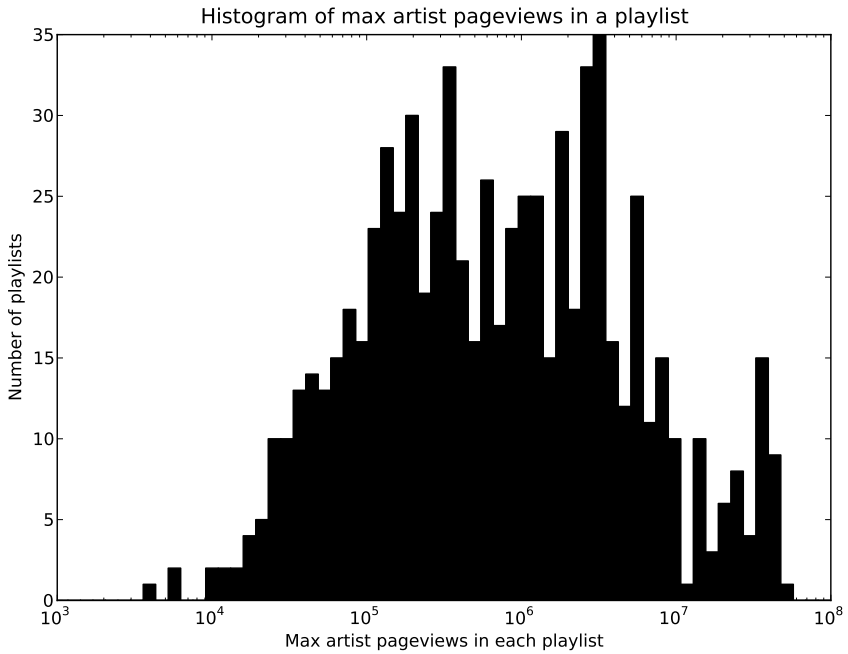


Figure 4.14: Histogram of max artist pageviews in a playlist

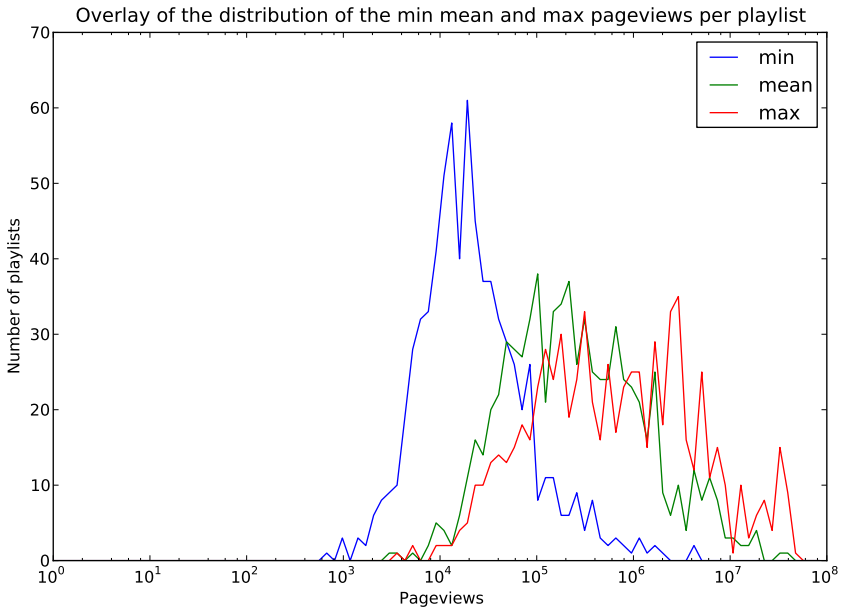


Figure 4.15: An overlay of the distribution of the minimum, mean and maximum pageviews per playlist

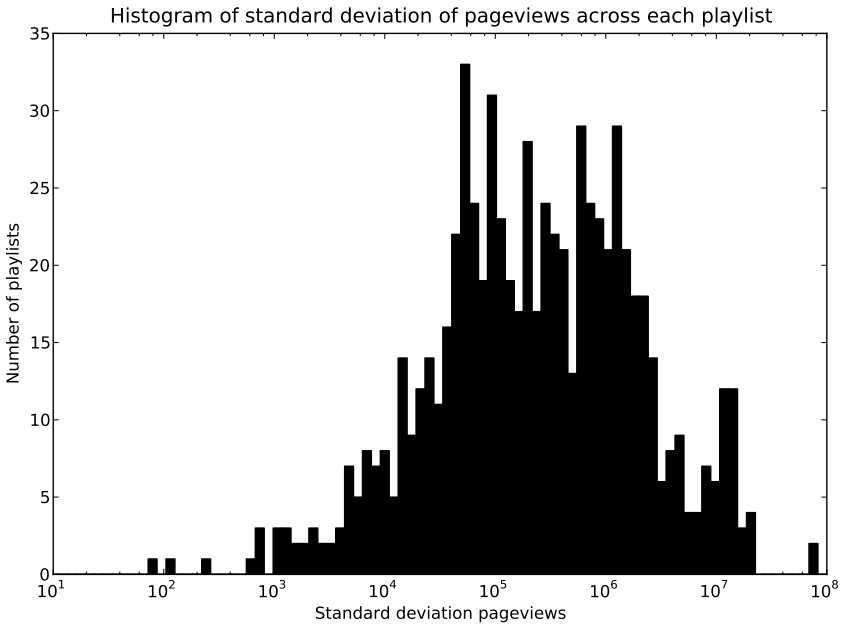


Figure 4.16: A histogram of the squareroot of the variance of each playlist's artists pageviews, with outliers ignored (squareroot of the variance $> 10^7$)

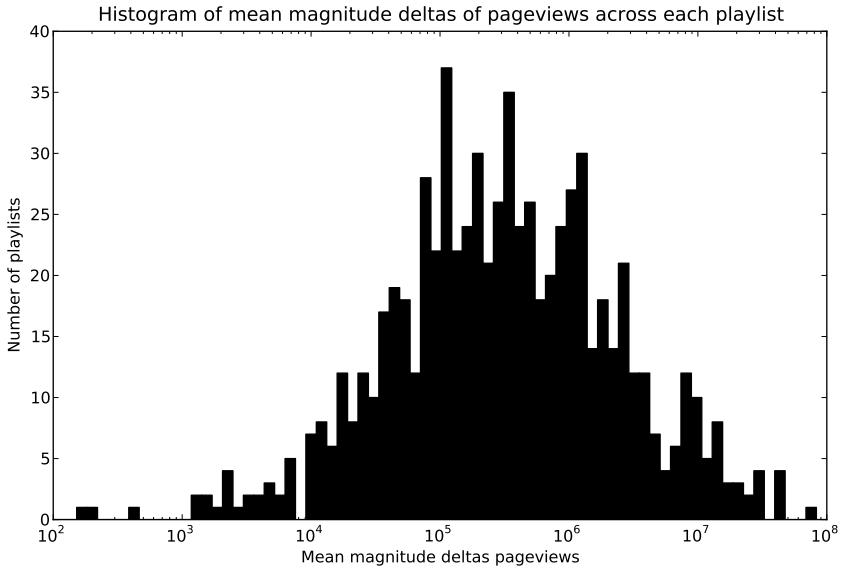


Figure 4.17: A histogram of the mean magnitude deltas of each playlist's neighbouring artists pageviews

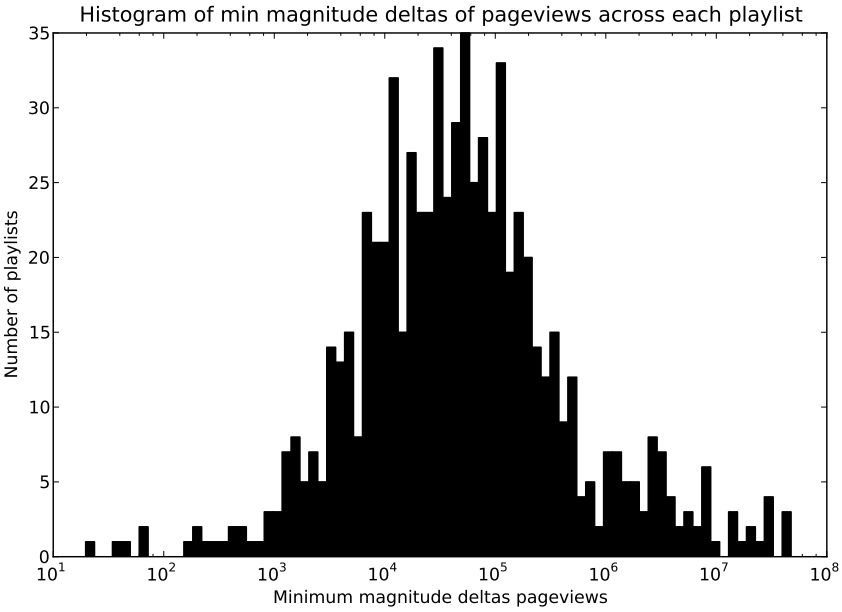


Figure 4.18: A histogram of minimum magnitude deltas of pageviews across each playlist

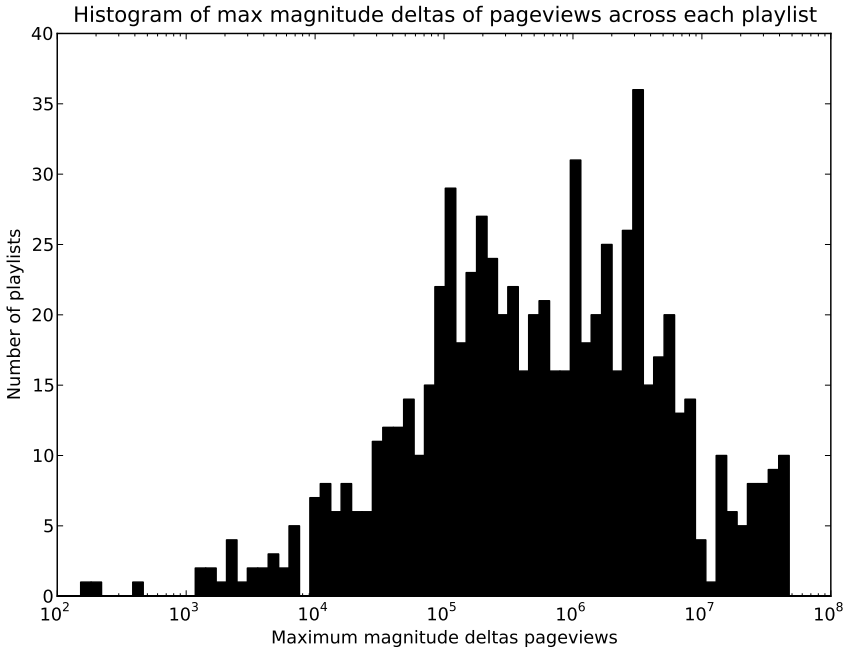


Figure 4.19: A histogram of maximum magnitude deltas of pageviews across each playlist

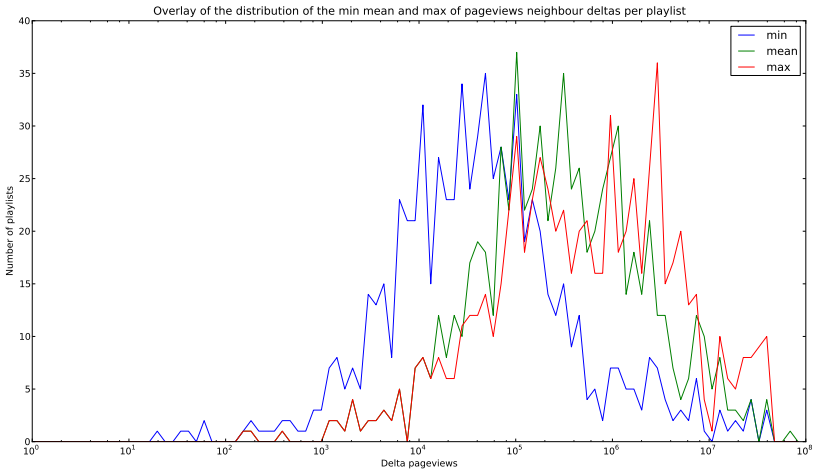


Figure 4.20: An overlay of the distribution of the minimum, mean and maximum magnitude delta pageviews per playlist

derlying dataset's degree distribution (Section 3.3.2). That is, playlists are composed mostly of songs from artists with moderate pageviews (around 100,000) with occasional material from artist with considerably higher pageviews (two to three orders of magnitude).

4.6 Discussion

After detailing ways to generate better input from users and better keep their interest, the idea of the web as a platform was briefly discussed. An interactive model for a web-based, radio-style group recommender system was then specified. The system itself was described, including a conceptual overview, the user interface and the core backend. Finally, the playlists produced by the system while gathering feedback from users were analysed and evaluated.

One of the goals behind the SoSoRadio system is to provide a means for communities of listeners to form. While the prototype has only one station, a means of self-organisation is possible simply by having multiple instances running in parallel. In this case, when a user goes to the site, they would be presented with a selection of stations, seeing a visual representation of what is currently playing on each of these stations. Through this initial process a user places themselves in a community of listeners. If the user's needs aren't being met – they are consistantly being out-voted – they can change to a different station, and so to a different community.

While the request aggregation was deliberately left uncomplicated for the prototype, there are other approaches that could be used there which could improve overall listener satisfaction or minimize the number of users who become disgruntled over time by being repeatedly out-voted. The most notable of these approaches is the group optimisation described in Baccigalupo [2009].

In order to make a complete, automated, playlist-broadcasting system, some improvement is needed to the current standard automatic crossfade. Some amount of phrase segmentation and alignment of neighboring tracks on certain types of playlists has the potential to vastly improve subjective response to computationally generated playlists' automatic playback. While such a mechanism does not currently exist within SoSoRadio the theoretical underpinning is discussed in Appendix A.

While some analysis is discussed with regard to the playlists produced by this system, this process brings to light the lack of robust means of evaluating playlists. One particular hindrance is the lack of methods to compare one playlist to another. With such a tool neighboring session playlists could be compared in much the same way as neighbouring songs. This would allow personalisation at the level of playlist via for example the propagation of sequences

of ratings to similar playlists (or subsets of playlists) analogous to the genre label propagation used by Sordo et al. [2007]. In order for such a technique to be possible, it is necessary to specify a means to compare playlists and quantify their difference.

Chapter 5

A Method to Describe and Compare Playlists

"For time is the longest distance between two places."
–Tennessee Williams, *The Glass Menagerie*, 1944

5.1 Introduction

Inherent to the design of any recommender or retrieval system is a means of display or delivery of selected content. For a system that recommends music this means the playback of an audio file. Listening to or playing a piece of music takes time dependent on the duration of that recording. Given this link between music and time, when considering what information is relevant for a recommendation it is vital to consider the context of time; that is, what music has been played before or will be played after the current recommended song. Yet little is understood about how playback order affects the success or failure of a recommendation of a piece of music. Whether a system makes user-based, object-based or hybrid recommendations, a better awareness and use of playback order will yield an improved music recommender system.

In order to take advantage of the effect of playback order, it is necessary to have some means of comparing playlists with one another. While ratings-based generic recommender strategies could be employed, such techniques could only be used in systems that allow for the rating of playlists directly (as opposed to the much more common rating of member songs). Alternatively, a distance measure between playlists can be used to facilitate the prediction and generation of well-ordered lists of song sequences for recommendation. This has the advantage of being applicable to the vast majority of existing playlist generation systems, many of which do not collect playlist level ratings from their users. Further, a measure of playlist distance has a number of other applications in music recommender and discovery systems including label propagation, predictive personalization and context tuning to name a few.

We propose an objective distance measure between playlists. To better understand why such a measure is needed, Section 5.2 revisits background information in existing playlist generation and evaluation techniques. While any sufficiently expressive and low-dimensional feature is compatible with our playlist measure, we use a novel social tag-based feature we have developed for this research. This song-level feature is detailed in Section 5.3. This is followed by an explanation of our distance measurement itself in Section 5.4. Putting this into practice, we detail some proof-of-concept evaluation in Section 5.5. We discuss the results of this evaluation and possible extensions in Section 5.6.

5.2 Playlist as Delivery Mechanism

In this section we survey the use of playlists for the delivery of content in existing recommendation and retrieval systems. This is followed by a review of current evaluation methods for generated playlists. These two surveys will show both the widespread use of playlist generation in music recommendation and discovery systems and the need for more quality evaluation of these systems.

While this brief survey is focused on automatic playlist generation, there is a wealth of both academic and lay work discussing various aspects manual human-driven playlist construction that may be of interest to the reader. Work in this area tends to deal with radio [Ahlkvist and Faulkner, 2002] or club and dance disc jockeys [Brewster and Broughton, 2006], being the two principal areas where the explicit construction of ordered lists of songs are tied to the field. It is with these areas of manual playlist construction in mind that we will examine past efforts in both automatic playlist construction and evaluation techniques.

5.2.1 Usage in the Wild

There have been many music recommendation and retrieval systems that employ some kind of automatic playlist construction within their system. Frequently this is done as a means of content delivery or, less often, as a way of facilitating human evaluation of an underlying process such as content-based music similarity or recommendation. What follows is a brief survey of existing methods of playlist generation both with and without human intervention.

Hayes and Cunningham [2000] details a web-based system for personalized radio. In this early system users create and publish playlists facilitated through a process analogous to collaborative filtering. This results in quasi-automatic playlist creation, with any sequence ordering depending entirely on the user. O'Hara et al. [2004] describes another variation of the social interaction intermediary, the *Jukola* system. This system creates playlists via democratic vote on every song using mobile devices of listeners in the same physical space. Liu

and Reimer [2008] Furthers the idea of collaborative human generation, via a system called *Social Playlist*. This system is based on the idea of social interaction through playlist sharing, integrating mobile devices and communal playback.

Aucouturier and Pachet [2002] describes a fully-automatic rule-based system. This system uses existing metadata such as artist name, song title, duration and beats per minute. The system is designed from the ground up to be scalable and is shown to work given a database of 200,000 tracks. Avesani et al. [2002] takes an approach that is derived from recommender systems. Here the authors use the ratings and personalization information to derive radio for a group. Platt et al. [2002] shows an attempt to optimize a playlist based on known user preference as encoded in song selection patterns. This effort uses Gaussian process regression on user preference to infer playlists. The system uses existing a priori metadata as the features for selection. Knees et al. [2006] uses web-mining-derived artist similarity with content-based song similarity to generate playlists automatically. This system combines these two spaces to minimize the use of signal analysis. A byproduct of this optimization is improved playlist generation, as is shown in a small evaluation with human listeners.

Baccigalupo [2009]; Baccigalupo and Plaza [2007] details the *Poolcasting* system. Poolcasting uses dynamic weighting of user preferences within a group of users who are all listening to a common stream with the goal of minimizing displeasure across the entire group. This results in a system that is very similar to popular commercial radio in terms of its output. A method for created playlists using an artist social graph, weighted with acoustic similarity is shown in Fields et al. [2008b]. This method takes a start and end song and constructs a playlist using maximum flow analysis on the weighted graph. Another technique for playlist construction based on the selection of paths between the start and end songs is shown in Flexer et al. [2008]. In this system content-based similarity is used to project a set of songs onto a 2-D map, then a path is found from the start song to the end song with the goal of minimizing the step size between each member song. A recent approach uses co-occurrence in n-grams extracted from the internet radio station Radio Paradise[1] to deform a content-based similarity space Maillet et al. [2009]. This deformed space is then used in a manner that is similar to Flexer et al. [2008] to generate paths from one song to another, minimizing step distance throughout the path.

Also of note is Ragno et al. [2005], which in contrast to most of the previous systems, uses nearest neighbour co-occurrence in radio playlist logs to

[1] http://radioparadise.com

determine song similarity. While the evaluation was preliminary this method shows promise.

5.2.2 Evaluation Methods

The most prevalent method of evaluation used in playlist generation systems is direct human evaluation by listening, as is discussed in Section 2.8.3. The system detailed in Pauws and Eggen [2002], a rule-based automatic playlist generator that uses features derived from metadata, is similar to Aucouturier and Pachet [2002]; Platt et al. [2002]. Of note in Pauws and Eggen [2002] is the thorough human listener testing which shows the automatic playlist generator performing considerably better than songs ordered randomly. This evaluation, though better than most, still fails to compare the automatic playlists against human expert playlists. Additionally, to reduce test time, the evaluation uses arbitrary one-minute clips from the songs rather than the entirety of the song or an intentionally chosen segment. A content-based similarity playlist generator with a novel evaluation is seen in Pampalk et al. [2005]. Here the authors track the number of times the user presses the *skip* button to move on from the currently playing song. All songs that are skipped are considered *false positives* and those that are completely played are treated as *true positives*. From this, many standard information retrieval techniques can be used in the evaluation, resulting in a rich understanding of the results. Ultimately, it is still human user listening evaluation though and its biggest drawback is playback time. Assuming an average song length of three minutes it would take an an hour (per listener) to listen to 20 songs plus additional time for listening to songs that are ultimately skipped. This skip-based evaluation framework is further used in Bosteels et al. [2009] where existing last.fm user logs (which include skip behavior) are analyzed using fuzzy set theory to determine playlist-generation heuristics in the system. Additionally, many systems of playlist generation lack formal evaluation all together.

5.2.3 Summary

While a number of techniques have been employed to create playlists for a variety of functions, there exist few for the evaluation of the generated playlists. These evaluation techniques rely heavily on time-consuming human evaluation. Beyond that, no means of objectively comparing playlists with one another has yet been published. In Section 5.4 we will propose just such a means. First we will describe a novel song-level feature based on tags. A tag-based feature will encode socio-cultural data that is missing from analogous content-based features, though social tags bring about some other problems.

Figure 5.1: The tag cloud for Bohemian Crapsody by Sickboy, from Last.fm.

5.3 Topic-Modelled Tag-Clouds

In order to encode playlists in a low-dimensional representation we must first represent their member songs in as a low-dimensional vector. Here we use a Topic-Modelled Tag Cloud (TMTC) as a pseudo-content-based feature, in a way that is functionally analogous to various pure content-based methods. Using tags and topic models in this way is novel and what follows is an explanation of the process of building this feature.

5.3.1 Tags as Representation

A *tag* is a word or phrase used to describe a document of some kind, typically on the Web. Various kinds of documents are described using tags on the Web including photos[2], videos[3] and music[4]. An aggregated collection of tags, weighted by the number of users who ascribe it to a given object, is commonly referred to as a *tag cloud*.

Tag clouds get their name from the most common visualization method used with them, where each tag is displayed with the font size in proportion to the weight, arranged in a way that resembles a cloud. An example of a tag cloud[5] can be seen in Figure 5.1 As can be seen in this example, tag clouds provide a rich description of the music it describes. Tags and collections of tags in various forms provide the basis for many techniques within music informatics including recommendation, retrieval and discovery applications [Aucouturier and Pampalk, 2008; Lamere, 2008].

In addition to human-generated tags being used, there is some research directed toward the automatic application of tags and inference of associated weights on unlabelled pieces of music [Barrington et al., 2008; Bertin-Mahieux et al., 2008; Eck et al., 2007; Hoffman et al., 2009].

[2] e.g. http://flickr.com
[3] e.g. http://youtube.com
[4] e.g. http://last.fm or http://musicbrainz.org
[5] This tag cloud is for the track Bohemian Crapsody by the artist Sickboy. The tags and the rendering both come from last.fm, available at http://www.last.fm/music/Sickboy/_/Bohemian+Crapsody/+tags

5.3.2 Reducing the Dimensionality

There exist some techniques [Begelman et al., 2006, for example] to determine semantic clustering within a tag cloud; however, these systems are built to facilitate browsing and do not create a representation with a sufficiently reduced dimensionality/number of dimensions. The previous work of Levy and Sandler [2008] comes the closest to the dimensional reduction required, also dealing with social tags for music. This work, through the use of aspect models and latent semantic analysis, brings the dimensionality down into the hundreds, while preserving meaning. But, this order of dimensions is still too high to compute meaningful distance across multi-song playlists. A feature with a number of dimensions of the order 10^2 would suffer from the curse of dimensionality [Weber et al., 1998]: because of its high dimensionality, any attempt to measure distance becomes dominated by noise. However, a technique developed for improved modelling in text information retrieval, *topic models* provide the reduced dimensional representation we require. Topic models are described in Blei and Lafferty [2009] as "probabilistic models for uncovering the underlying semantic structure of [a] document collection based on a hierarchical Bayesian analysis of the original text." In topic modeling, a *document* is transformed into a *bag of words*, in which all of the words of a document are collected and the frequency of the occurrence in recorded. We can use the weighted collection of tags in a tag cloud as this bag of words, with tags serving as tokenized words.

There are a few different ways of generating topic models; for our feature generation we will be using latent Dirichlet allocation [Blei et al., 2003], treating each tag cloud as a bag-of-words. In LDA, documents (in our case tags clouds of songs) are represented as a mixture of implied (or *latent*) topics, where each topic can be described as a distribution of words (or here, tags). More formally given the hyper-parameter α, and the conditional multinomial parameter β, Equation 5.3.2 gives the joint topic distribution θ, a set of N topics $\mathbf{z}$ and a set of M tags $\mathbf{w}$.

$$p(\theta, \mathbf{z}, \mathbf{w}|\alpha, \beta) = p(\theta|\alpha) \prod_{n=1}^{N} p(z_n|\theta)p(w_n|z_n, \beta) \tag{5.1}$$

In Figure 5.2 LDA is shown as a probabilistic graphical model. In order to create topic models using LDA, we need to specify $p(\theta|\alpha)$ and $p(z_n|\theta)$. We estimate our parameters empirically from a given corpus of tag clouds. This estimation is done using *variational EM* as described in Blei et al. [2003]. This allows topic distributions to be generated in an unsupervised fashion, though the number of topics in a corpus must be specified a priori.

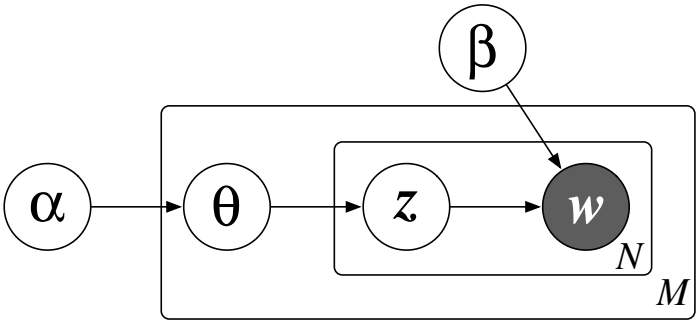


Figure 5.2: The graphic model of LDA Blei et al. [2003]. The replicates are represented as the two boxes. The outer box M represents the corpus of documents, while the inner box N represents the repeating choice of topics and words which make up each document.

Once the LDA model is generated, it is used to infer the mixture of topics present in the tag cloud for a given song. This is done via *variational inference* which is shown in Blei et al. [2003] to estimate the topic mixture of a document by iteratively minimizing the KL divergence from variational distribution of the latent variables and the true posterior $p(\theta, \mathbf{z}|\mathbf{w}, \alpha, \beta)$.

This process in its entirety is shown as a block diagram in Figure 5.3. Once it is completed for every song in our dataset, we will have a single vector with a dimensionality equal to the number of topics in our LDA whose entries indicate topic occupancy for that song.

5.4 Playlists as a Sequence of Topic Weights

Given the single-vector-per-song reduction, we represent the playlists these song are in as ordered sequences of these vectors. Thus each playlist is represented as an $l \times d$-dimensional vector, where l is the number of songs in a given playlist and d is the number of topics in our LDA model.

5.4.1 Measuring Distance

To both manage and measure the distance between these $l_i \times d$ dimensional vectors we use audioDB[6]. The use of audioDB to match vectors of this type is detailed in Rhodes et al. [2011]. Briefly, distance is calculated by means of a multidimensional Euclidian measure.

$$d^2_{Euc}(x, y) = \sum_{i=1}^{l} \sum_{j=1}^{d} (x_{ij} - y_{ij})^2 \tag{5.2}$$

Here l_i is an arbitrary length subsequence of i vectors. In practice, i is selected to be less than or equal to the smallest sequence length for a complete playlist

[6]source and binary available at http://omras2.doc.gold.ac.uk/software/audiodb/

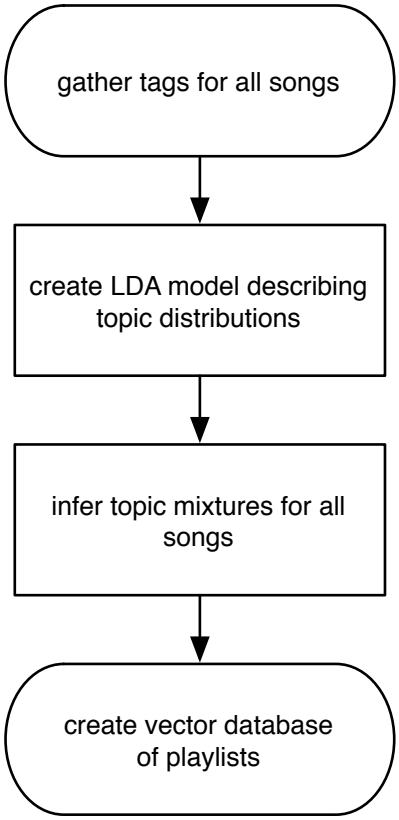


Figure 5.3: The complete process for construction of a TCTM feature set

in a dataset. The distance between two playlists is then the minimum distance between any two length i sub-vectors drawn from each playlist. One effect of this technique is easy handling of playlists of unequal length.

This type of distance measurement has been used with success on sequences of audio frames [Casey et al., 2008a; Casey and Slaney, 2006]. The distance measure in use between vectors can also be changed. In particular there has been work showing that statistical features (such as topic models) may benefit from the use of Manhattan distance [Grauman and Darrell, 2004; Howarth and Rüger, 2005], however for our prototypical evaluation we have used simple Euclidean distance.

5.5 Evaluation

The goal of our evaluation is to show the fitness of our distance measurement through preliminary retrieval tests: searching for playlists that start at the same time of day as our query playlist and searching for the playlists from the same station from a database of stations of the same genre. We examine the logs of a large collection of radio stations, exhaustively searching example sets. Through precision and recall we see that our measure organizes playlists in a

Source	S_t	S_{mt}	P_t	$P_{avg(time)}$	$P_{avg(songs)}$
Whole yes.com	885810	2543	70190	55min	12.62
"Rock" stations	105952	865	9414	53min	11.25
"Jazz" stations	36593	1092	3787	55min	9.66
"Radio Paradise"	195691	2246	45284	16min	4.32

Table 5.1: Basic statistics for both the radio log datasets. Symbols are as follows: S_t is the total number of song entries found in the dataset; S_{mt} is the total number of songs in S_t where tags could not be found; P_t is total number of playlists; $P_{avg(time)}$ is the average runtime of these playlists and $P_{avg(songs)}$ is the mean number of songs per playlist.

predictable and expected way.

5.5.1 Dataset

In order to test these proposed techniques a collection of radio station logs were gathered. These logs come from a collection of broadcast and online stations gathered via Yes.com[7]. The logs cover the songs played by all indexed stations between 19-26 March 2010. For our evaluation task using this data source we look at subsets of this complete capture, based on genre labels applied to these stations. Specifically we examine stations of the genres *rock* and *jazz*. The complete Yes.com dataset also includes stations in the following genre categories: *Christian*, *Country*, *Electronica*, *Hip-Hop*, *Latin*, *Metal*, *Pop*, *Punk*, *R&B/Soul*, *Smooth Jazz* and *World*. These labels are applied by the stations themselves and the categories are curated by Yes.com. Additionally, the play logs from Radio Paradise[8] from 1 January 2007 to 28 August 2008 form a second set. We then attempted to retrieve tag clouds from Last.fm[9] for all songs in these logs. When tags were not found the song and its associated playlist were removed from our dataset.

These logs are then parsed into playlists. For the radio logs retrieved via the Yes API, the top of every hour was used as a segmentation point to approximate the boundary between distinct programs. We assume that program are more likely than not to start and finish on the hour in US commercial broadcast. Note that this method of boundary placement will almost certainly over-segment radio programs, as many radio programs are longer than one hour. However, given that our distance measure compares fixed-length song sequences across playlists, this over-segmentation should produce only minimal distortion in our results. The Radio Paradise logs include all the *links* or breaks between songs

[7] http://api.yes.com
[8] http://www.radioparadise.com/
[9] http://last.fm

where the presenter speaks briefly. For experiments using the Radio Paradise logs these links are used as playlist boundaries. This leads to a slight difference in the type of playlist used from Radio Paradise versus Yes. The playlists coming from Radio Paradise represent strings of continuously played songs, with no breaks between the songs in the playlists. The playlists from Yes are approximations of a complete radio program and can therefore contain some material inserted between songs (e.g. presenter link, commercials).

Statistics for our dataset can be seen in Table 5.1. We then use the tags clouds for these songs to estimate LDA topic models as described in Section 5.3[10]. For all our experiments we specify 10 topic models a priori. The five most relevant tags in each of the topics in models trained on both the "Rock" and "Jazz" stations can be seen Table 5.2.

5.5.2 Daily Patterns

Our first evaluation looks at the difference between the time of day a given query playlist starts and the start time for the closest n playlists by our measure. For this evaluation we look at the 18-month log from Radio Paradise as well as the "Rock" and "jazz" labelled stations from Yes.com, each in turn. Further we use a twelve-hour clock to account for overnight recorded program loops. The basis for this test relies on the hypothesis that for much commercial radio content in the United States, branding of programs is based on daily repeatable of tone and content for a given time of day. It should therefore be expected that playlists with similar contours would occur at similar times of day across stations competing for similar markets of listeners.

Figure 5.4 shows the mean across all query playlists of the time difference for each result position for the closest n results, where n is 200 for the Radio Paradise set and 100 for the Yes.com set. The mean time difference across all three sets is basically flat, with an average time difference of just below 11000 or about three hours. Given the maximum difference of 12 hours, this result is entirely the opposite of compelling, with the retrieved results showing no correspondence to time of day. Further investigation is required to determine whether this is a failure of the distance metric or simply an accurate portrayal of the radio stations logs. A deeper examination of some of the Yes.com data shows some evidence of the latter case. Many of the playlist queries exactly match (distance of 0) with the entirety of the 200 returned results. Further these exact match playlists are repeated evenly throughout the day. One of

[10]Our topic models are created using the open source implementation of LDA found in the gensim python package available at http://nlp.fi.muni.cz/projekty/gensim/ which in turn is based on Blei's C implementation available at http://www.cs.princeton.edu/~blei/lda-c/

station label	t_1	t_2	t_3	t_4	t_5
	Snow Patrol	Bob Marley	female vocalists	aupa Pete	80s
	rumba	Feist	Anna Nalick	whistling	new wave
"Rock"	90s	john mayer	Chicas	Triple J Hottest 100	david bowie
	green day	drunk love	playlist 2009	review	neuentd
	Dynamit	feist backing vocals	Sarah McLachlan	fun as fuck	synth pop
	motown	john mayer	60s	Sade	Flamenco
	soul	acoustic	jazz - sax	deserves another listen	tactile smooth jazz
"Jazz"	70s	corinne bailey rae	acid jazz	till you come to me	guitar ponder
	funk	bonnie raitt	reggae	piano	cafe mocha
	Disco	David Pack 2	cool jazz	2010	wine

station label	t_6	t_7	t_8	t_9	t_{10}
	classic rock	TRB	reminds me of winter	Needtobreathe	Krista Brickbauer
	60s	ElectronicaDance	kings of leon	plvaronaswow2009	day end
"Rock"	70s	mysterious	songs that save my life	The Script	i bought a toothbrush
	The Beatles	best songs of 2009	songs to travel	brilliant music	bluegrass
	the rolling stones	tribute to george	Muse	van morrison	omg
	follow-up	rnb	female vocalists	classic rock	Smooth Jazz
	jazz	soul	norah jones	80s	saxophone
"Jazz"	instrumental	female vocalists	dido	rock	smooth jazz sax
	guitar	Neo-Soul	jazz	70s	contemporary jazz
	latin jazz	Robin Thicke	vocal jazz	yacht rock	instrumental

Table 5.2: The five most relevant tags in each topic. Upper model is all the Yes.com Rock stations, lower model is all Yes.com Jazz stations.

these queries is shown in Figure 5.5. The existence of these repeating playlists throughout the day ensures this task will not confirm our hypothesis, perhaps due to programming with no reliance on time of day at least in the case of Radio Paradise.

5.5.3 Inter-station vs. Intra-station

In this evaluation we examine the precision and recall of retrieving playlists from the same station as the query playlist. Here we look at the "Rock" and "Jazz" labelled stations retrieved via the Yes API, each in turn. It is expected that a given station will have its own *tone* or particular *feel* that should lead to playlists from that station being more apt to match playlist from their generating station then with other stations from the same genre. More formally, for each query we treat returned playlists as relevant, true positives when they come from the same station as the query playlist and false positives otherwise. Based on this relevance assumption, precision and recall are calculated using the following standard equations.

$$P = \frac{|\{relevantplaylists\} \bigcap \{retrievedplaylists\}|}{|\{retrievedplaylists\}|} \tag{5.3}$$

$$R = \frac{|\{relevantplaylists\} \bigcap \{retrievedplaylists\}|}{|\{relevantplaylists\}|} \tag{5.4}$$

The precision versus recall for a selection of stations' playlists from both the "Rock" and "Jazz" stations are shown in Figure 5.6. When considering the precision and recall performance it is useful to compare against random chance retrieval. There are 100 stations labeled "Rock" and 48 labeled "Jazz". Under chance retrieval a precision of 0.01 would be seen for "Rock" and 0.0208 for "Jazz".

5.5.4 Summary

Two different evaluation tasks have been run using real-world radio-log data to explore the usefulness of our playlist match technique. The first of these, an examination of the time difference was flat across result-length variance. While this implies lack of discrimination into daily patterns, it is not possible to determine from the available data whether this is an accurate reflection of the progamming within the dataset or a result of the distance measure not being sufficient for the task. The second task shows the performance of retrieving hourly playlists from a selection of stations using playlists from that station as a query. Here we see a great deal of promise, especially when comparing the query results against random chance, which it outperforms considerably.

5.6 Discussion

Having reviewed recent work in various methods of playlist generation and evaluation in Section 5.2, we have shown that there is a need for better ways to objectively compare playlists to one another. We detailed a method of doing so in Section 5.4, though first, to better encode socio-cultural data along with content-based data, we presented a novel tag-based feature, TMTC, using tags summarized using LDA topic models in Section 5.3. This was follow by two task evaluations to examine our playlist-matching technique and song feature on real-world playlist data from radio logs in Section 5.5.

While our evaluation shows the promise of this technique on sampled data, there is much room for improvement. Principal among these is the exploration of non-Euclidean distance measures. Manhattan distance (or L_1) seems to have the most direct applicability and its use could prove to be quite beneficial. Another area for future work is in the use of the measure on further data and datasets. One of the best ways to improve here would be in the use of datasets with a more exactly known ground truth, in order to best apply known recommender and retrieval evaluation methods to them.

This leads to a further avenue of future work, testing the measure against direct human evaluation. While our matching technique has many uses with recommendation and discovery, if it proved to align with human evaluation it would be considerably more useful.

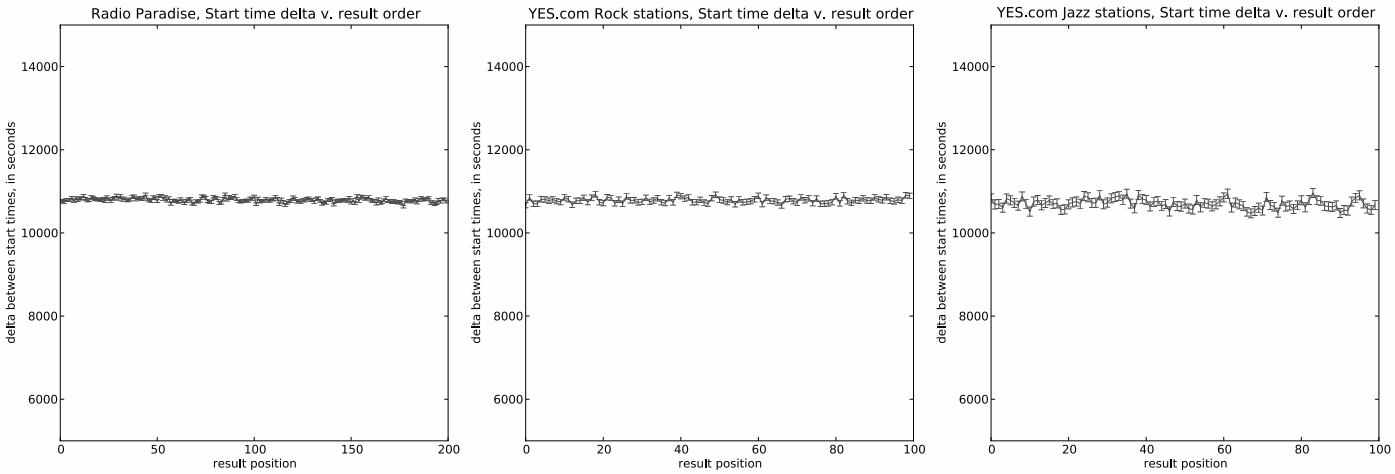


Figure 5.4: The mean start time difference, with squared error of the mean.

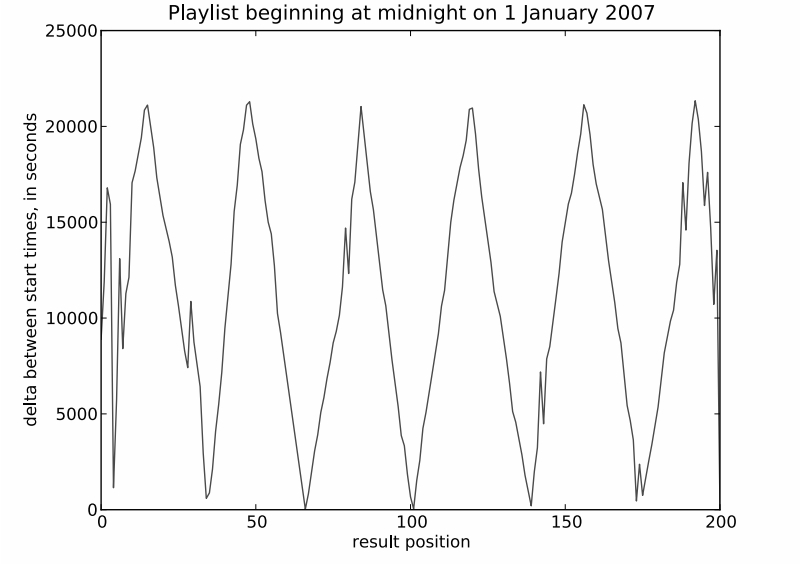


Figure 5.5: The time of day difference from the query playlist for 200 returned results, showing even time of day spread. Note that all the results show here have a distance of 0 from the query.

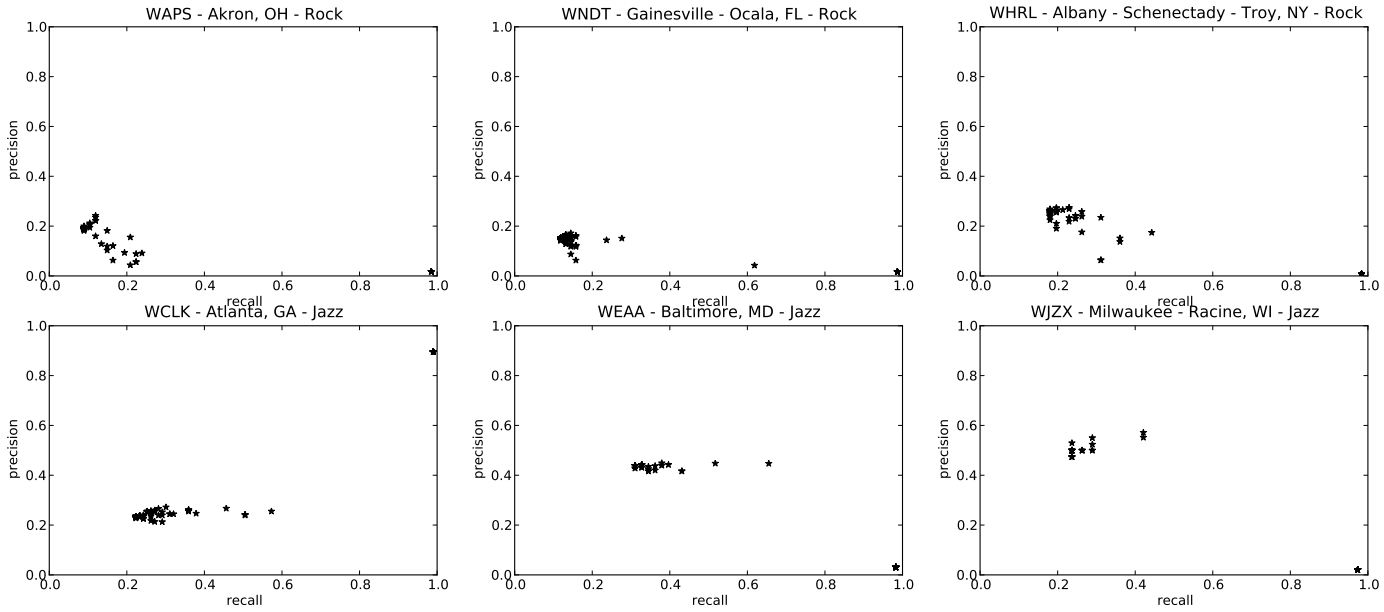


Figure 5.6: Precision versus Recall for six stations when using their hourly playlists to query for other playlists from the same station. In each query the number of results retrieved is selected to maximize the F1 score.

Chapter 6

Conclusions

> "Until I die there will be sounds. And they will continue following my death. One need not fear about the future of music."
> – John Cage, *Experimental Music*, 1957

6.1 Summary

In this thesis we investigated the playlist, a set of songs to be listened to together, typically in a specified order. We surveyed existing playlisting methods and the automatic similarity estimation techniques many of them depend on. We proposed a novel multimodal similarity measures integrating content-based similarity with artist relational social graphs. We then used this complex similarity and community segmentation to drive a user-steerable automatic radio station. In attempting to evaluate this prototypical application, we specified a new means of comparing playlists, based on a novel low-dimensional song level feature using social tag descriptors. In its totality this work has significantly improved the state of the art in the understanding and construction of playlists through betterment of these composite parts.

6.2 Contributions

1. A thorough review of previous research work related to playlists, across multiple disciplines. This show the interlink between music similarity and playlist generation, as well as an over-dependance on homogeneity as a marker of success in playlist construction. (Sections 2.5 and 2.8)

2. A published open dataset, sampled from Myspace, of artists, their songs and their relationships to other artists; the complex network analytics showing that this sample is similar to other webs of data. (Chapter 3)

3. Analysis using the mutual information across multiple distance measures, both social and acoustic, within the sampled myspace set which shows very little (less than 1 bit) of shared information between social and

acoustic distance (*i.e.* Do you sound like your friends? No more than anyone else.) (Chapter 3)

4. The application of graph path-finding techniques as a novel means for playlist generation, specifically the use of maximum-flow analysis. (Chapter 3)

5. A system model that provides a flexible framework for experimentation in the use of a weighted-graph-based datasets into a user-centric group recommender system. (Chapter 4)

6. A low-dimensional vector representation of a song, using weighted social tags and latent Dirichlet allocation (Section 5.3). Results showing that quantifying dissimilarity between playlists by representing a playlist's constituent songs in a low dimensional space and applying sequence matching techniques is an effective means to compare playlists as seen in the ability to retrieve playlists from the same radio station from a large set. (Section 5.4)

6.3 Limitations and Future Work

The work presented in this thesis poses as many questions as it answers. In this section we discuss a few of these questions and possible avenue of research. In part these questions stem from the limits of our research. In particular: the requirement to have full knowledge of an artist network, and the related lack of scale; the reliance on computationally complex audio-derived features; and the lack of a satisfying and pragmatic solution to the particular issues surrounding the evaluation of playlists.

Understanding Network Ecology

When considering multi-modal analysis of networks of musicians and their music there are a number of open avenues for continued exploration. It remains to examine community detection methods that operate locally, without knowledge of the entire network. We also plan to address further directed-artist-graph analysis, bipartite networks of artists and listeners, different audio-analysis methods, and the application of these methods to music recommendation.

Many of these tasks require the expansion of our sample network. The goal of any effort to expand the sample size drawn from a network such as Myspace is best focused on ways to make the sample set more indicative of the whole. While it is impossible to assess this without capturing the entire graphs some assumptions can be made. Snowball sampling has a tendency to oversample hubs. Given this, a better expanded network is likely to result

through the selections of new starting seed artist (most likely at random) and proceeding via a breadth-first crawl until that crawl results in overlap with the known network. It is reasonable to assume that this method, when used over multiple hubs, will produce a lower proportion of high centrality hubs then simply continuing further with the existing breadth first crawl. With a lower proportion of these over-sampled hubs, the social structure of the sample would better match that of the whole.

Use of Local-Only Awareness

One of the most significant limitations of the playlist technique employed in SoSoRadio (Chapter 4), or any playlist-generation method based on the analytics of Chapter 3, is the dependency on a complex indexing process and an assumption that the indexing process has covered the whole graph. Among other problems, this creates a delay between the system and the data that can yield incorrect and out-of-date results. One way these limitations can be overcome is by developing algorithms that use *local-only awareness*, the direct surroundings of the query songs, as a starting point. We have developed a lightweight web application called Roomba Recon[1] to explore the feasibility of this approach. This application creates a playlist between any two songs available on Soundcloud[2], following the social links between artists in the network. A screen capture of the application is shown in Figure 6.1 and a description follows.

Blind paths

The notion of finding an optimal path from between two nodes breaks down when the whole graph is not known a priori. Rather, a search model is more helpful. In this prototype, A* search [Hart et al., 1968] is employed from both the start and end song. In this way the graph is sampled as the playlist solution is constructed, rather than in advance. This introduces some amount of error, as the playlist generated will not provably be the shortest path. However it is possible, perhaps even likely that this error will not lead to a noticeable reduction in the system's ability to meet the needs of users. Testing of this sort remains to be carried out.

Replacing Audio Features

Another stumbling block in the removal of the requirement for a prebuilt index is the complexity of signal-based audio features. The suggested A* search requires a cost function in order to determine the distance from a candidate song

[1] The application is available for demonstration at http://doc.gold.ac.uk/~map01bf/recon/playlist. The python source is available at https://github.com/gearmonkey/roombarecon

[2] http://soundcloud.com

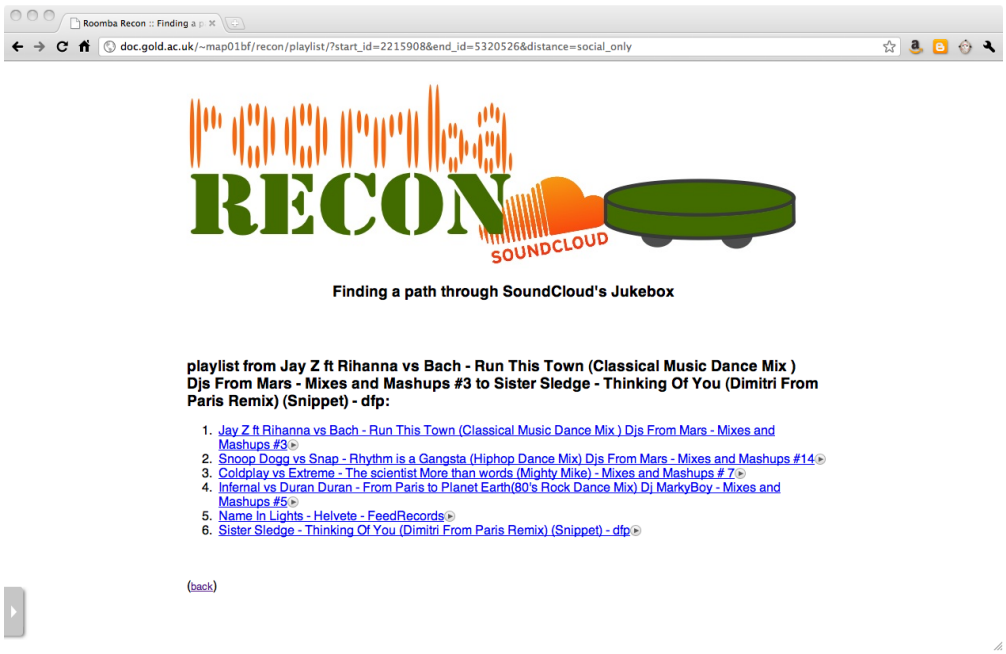


Figure 6.1: A playlist created by the Roomba Recon playlist generation system, creating a path between any two songs in the Soundcloud network without prior knowledge.

to the query song. While the use of content-based similarity measures seem obvious they are too slow to be practical in this context, where approximately 800 comparisons[3] are required to generate a playlist of length 10. Rather than rely on a spectral analysis of the signal, our Roomba Recon application allows for the use of a facsimile text-based feature inspired by the feature detailed in Section 5.3. The Soundcloud network allows artists to tag each song and other users to comment on the song in free text. By using a simple natural language processing technique, term frequency·independent document frequency (tf·idf) [Salton, 1989], on these comments, we can generate a reduced dimensional representation of these song, allow for the distance to be measured between pairs of songs. The speed tradeoff here is apparent. If the idf is calculated using a small random sample in advance, the term frequency for all songs in a comparison can be generated in a few minutes, rather than the few hours it would take to download and process the audio. What the differences are between these two pseudo-metric spaces, and whether there is a yet more-efficient feature space that can be used for this 'real-time'[4] playlist generation, remain open questions.

[3]This conservatively assumes each artist has 10 friends and 10 songs, actual use may require more song-to-song comparisons.

[4]When considering a playlist from the user's point of view, real-time generation means the next song in the playlist has to be selected and available prior to the end of the currently-playing song.

Improvements in Evaluation

While considerable effort has been made to perform robust evaluation on generated playlists and relatedly to compare differing similarity spaces, there are many possible ways to continue forward. While the best way to test playlists is through large-scale human evaluations, these are also the longest and most costly to run. New statistical methods offer other avenues of evaluation. The work of Herrera et al. [2010] adds time of day of listening using circular statistics and offers potential means of evaluation. Critically, any evaluation solution in the future will need to be multifaceted, using statistical and computational methods while also employing human evaluation through both listening and use testing.

6.4 Concluding Remarks

An underlying aim in this work is to show that playlist construction (and analysis) is intricately tied to understanding the relationships between songs. This is something that is both obvious and, in prior work, unacknowledged. Given that, it is no surprise that in order to meet the goal of making playlists richer, it is first necessary to broaden the scope of information encoded in descriptions of the similarity (or more generally, relationships) of pieces of music. In this work, this added information comes the inclusion of the social data of musicians along side the content they produce. This social data, while always important, has only recently existed in a form that allows for the scale of aggregation required. The improvements in playlist generation then followed from this more complex similarity measure, together with a shift in focus away from flat, homogenous playlists.

While this provides a clear improvement in the state of the art, it also raises many new and interesting questions. What other domains might offer better insight into music and music similarity? Just how necessary is musical content in making a high quality music recommendation, in the form of a playlist or something else? And if content doesn't help in making a music recommendation, what might that say about us? It has been said that humans are basically social animals, is it reasonable to assume we are social listeners?

Thank you for reading, and pick out another record would you, this one's nearly done.

Bibliography

Adomavicius, G. and Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, 17(6):734–749. (Cited on pages 36 and 95.)

Ahlkvist, J. A. and Faulkner, R. (2002). 'Will This Record Work for Us?': Managing Music Formats in Commercial Radio. *Qualitative Sociology*, 25(2):189–215. (Cited on page 122.)

Ahn, Y.-Y., Han, S., Kwak, H., Moon, S., and Jeong, H. (2007). Analysis of topological characteristics of huge online social networking services. In *The World Wide Web Conference (WWW)*, Alberta, Canada. (Cited on page 77.)

Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (1993). *Network flows: theory, algorithms, and applications*. Prentice-Hall, Inc. (Cited on page 73.)

Alghoniemy, M. and Tewfik, A. (2001). A network flow model for playlist generation. In *IEEE International Conference on Multimedia and Expo (ICME)*. (Cited on page 73.)

Amaral, L. A. N., Scala, A., Barthélémy, M., and Stanley, H. E. (2000). Classes of small-world networks. In *Proceeding of the National Academy of Sciences*. (Cited on page 78.)

Anand, S. S. and Mobasher, B. (2007). *Contextual Recommendation*, pages 142–160. Springer. (Cited on page 23.)

Andrieu, C., de Freitas, N., Doucet, A., and Jordan, M. I. (2003). An Introduction to MCMC for Machine Learning. *Machine Learning*, 50(1-2):5–43. (Cited on page 30.)

Anglade, A., Tiemann, M., and Vignoli, F. (2007). Virtual communities for creating shared music channels. In *International Conference on Music Information Retrieval (ISMIR)*. (Cited on page 74.)

Asante, M. K. (2008). *It's Bigger Than Hip Hop: The Rise of the Post Hip Hop Generation*. St. Martin's Press. (Cited on page 27.)

Aucouturier, J.-J. and Pachet, F. (2002). Scaling up playlist generation. In *IEEE International Conference on Multimedia and Expo (ICME)*. (Cited on pages 55, 100, 102, 123 and 124.)

Aucouturier, J.-J. and Pachet, F. (2004). Improving timbre similarity: How high's the sky? *Journal of Negative Results in Speech and Audio Sciences*. (Cited on pages 30, 55 and 75.)

Aucouturier, J.-J. and Pampalk, E. (2008). Introduction-from genres to tags: A little epistemology of music information retrieval research. *Journal of New Music Research*, 37(2):87–92. (Cited on page 125.)

Avesani, P., Massa, P., Nori, M., and Susi, A. (2002). Collaborative radio community. In *International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH)*, pages 462–465. Springer. (Cited on pages 56 and 123.)

Baccigalupo, C. (2009). *Poolcasting: an intelligent technique to customise music programmes for their audience*. PhD thesis, Institut d'Investigació en Intelligència Artificial. (Cited on pages 119 and 123.)

Baccigalupo, C. and Plaza, E. (2007). Sharing and Combining Listening Experience: A Social Approach to Web Radio. In *International Computer Music Conference (ICMC)*, Copenhagen, Denmark. (Cited on page 123.)

Barkhuus, L. and Dey, A. (2003). *Is Context-Aware Computing Taking Control away from the User? Three Levels of Interactivity Examined*, volume 2864 of *Lecture Notes in Computer Science*, pages 149–156. Springer. (Cited on page 36.)

Barrington, L., Turnbull, D., and Lanckriet, G. (2008). Auto-tagging music content with semantic multinomials. In *International Conference on Music Information Retrieval (ISMIR)*. (Cited on page 125.)

Baur, D., Boring, S., and Butz, A. (2010). Rush: Repeated recommendations on mobile devices. In *International Conference on Intelligent User Interfaces (IUI)*, pages 91–100, New York, NY, USA. ACM. (Cited on page 45.)

Begelman, G., Keller, P., and Smadja, F. (2006). Automated Tag Clustering: Improving search and exploration in the tag space. In *Collaborative Web Tag-*

ging Workshop , Co-Located with the World Wide Web Conference (WWW). (Cited on page 126.)

Bello, J., Daudet, L., Abdallah, S., Duxbury, C., Davies, M., and Sandler, M. (2005). A tutorial on onset detection in music signals. *IEEE Transactions on Speech and Audio Processing*, 13(5):1035 – 1047. (Cited on page 159.)

Bello, J. P. (2003). *Toward the Automated Analysis of Simple Polyphonic Music: A Knowledge-based Approach*. PhD thesis, Department of Electronic Engineering, Queen Mary, University of London. (Cited on page 159.)

Berenzweig, A., Logan, B., Ellis, D. P. W., and Whitman, B. P. W. (2004). A large-scale evaluation of acoustic and subjective music-similarity measures. *Computer Music Journal*, 28(2):63–76. (Cited on pages 29, 52, 55 and 74.)

Bertin-Mahieux, T., Eck, D., Maillet, F., and Lamere, P. (2008). Autotagger: a model for predicting social tags from acoustic features on large music databases. *Journal of New Music Research*, 37(2):101–121. (Cited on page 125.)

Blei, D. and Lafferty, J. (2009). *Topic Models*. Text Mining: Theory and Applications. Taylor and Francis. (Cited on page 126.)

Blei, D. M., Ng, A., and Jordan, M. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022. (Cited on pages 126 and 127.)

Blow, C. M. (2009). Swan Songs? *The New York Times*, 159(213):A17. (Cited on page 16.)

Bogdanov, D., Serrà, J., Wack, N., and Herrera, P. (2009). From low-level to high-level: Comparative study of music similarity measures. In *International Workshop on Advances in Music Information Research (AdMIRe), Co-Located with the IEEE International Conference on Multimedia and Expo (ICME)*. (Cited on page 31.)

Bogdanov, D., Serrà, J., Wack, N., and Herrera, P. (2010). Hybrid music similarity measure. Music Information Retrieval Evaluation eXchange (MIREX) Abstract. (Cited on pages 31 and 33.)

Bollen, D., Knijnenburg, B. P., Willemsen, M. C., and Graus, M. (2010). Understanding choice overload in recommender systems. In *ACM International Conference on Recommender Systems (RecSys)*, pages 63–70, New York, NY, USA. ACM. (Cited on page 103.)

Bosteels, K., Pampalk, E., and Kerre, E. E. (2009). Evaluating and Analysing Dynamic Playlist Generation Heuristics Using Radio Logs and Fuzzy Set Theory. In *Conference of the International Society of Music Information Retrieval (ISMIR)*. (Cited on pages 124 and 158.)

Brewster, B. and Broughton, F. (2006). *Last Night A DJ Saved My Life; The history of the disc jockey*. Headline Book Publishing, London, United Kingdom, 2nd edition. (Cited on pages 23, 26, 27, 122 and 158.)

Bryk, A. S. and Weisberg, H. I. (1976). Value-added analysis: A dynamic approach to the estimation of treatment effects. *Journal of Educational and Behavioral Statistics*, 1(2):127–155. (Cited on page 104.)

Bull, M. (2006). *Investigating the Culture of Mobile Listening: From Walkman to iPod*, volume 35 of *Computer Supported Cooperative Work*. Springer. (Cited on page 27.)

Cano, P., Celma, Ò., Koppenberger, M., and Buldu, J. M. (2006). The topology of music recommendation networks. *Chaos: An Interdisciplinary Journal of Nonlinear Science*. http://arxiv.org/abs/physics/0512266v1. (Cited on pages 74 and 78.)

Cano, P., Koppenberger, M., and Wack, N. (2005). Content-based music audio recommendation. In *ACM International Conference on Multimedia (ACMMM)*, pages 211–212. (Cited on page 31.)

Casey, M., Rhodes, C., and Slaney, M. (2008a). Analysis of minimum distances in high-dimensional musical spaces. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(5):1015 –1028. (Cited on page 128.)

Casey, M. and Slaney, M. (2006). The importance of sequences in music similarity. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Toulouse, France. (Cited on page 128.)

Casey, M., Veltkamp, R., Goto, M., Leman, M., Rhodes, C., and Slaney, M. (2008b). Content-Based Music Information Retrieval: Current Directions and Future Challenge. *Proceedings of the IEEE*, 96(4):668–696. (Cited on pages 74 and 158.)

Chen, C.-h., Härdle, W., Unwin, A., Cox, M. A. A., and Cox, T. F. (2008). Multidimensional scaling. In *Handbook of Data Visualization*, Springer Handbooks of Computational Statistics, pages 315–347. Springer Berlin Heidelberg. 10.1007/978-3-540-33037-0_14. (Cited on page 59.)

Clauset, A., Newman, M. E. J., and Moore, C. (2004). Finding community structure in very large networks. *Physical Review E*, 70(6):066111. (Cited on pages 74, 79 and 80.)

Cliff, D. (1999). Hang the DJ: Automatic sequencing and seamless mixing of dance-music tracks. Technical Report 104, Hewlett-Packard Laboratories. (Cited on pages 40 and 41.)

Cliff, D. (2006). *hpDJ: An automated DJ with floorshow feedback*. Consuming Music Together. Springer. (Cited on pages 40 and 41.)

Clifford, S. (2007). Pandora's long strange trip. *Inc. Magazine*, 7(10). (Cited on page 48.)

Cohen, W. W. and Singer, Y. (1999). Context-sensitive learning methods for text categorization. *ACM Transactions on Infinite Systems*, 17(2):141–173. (Cited on page 23.)

Costa, L. F., Rodrigues, F. A., Travieso, G., and Boas, P. R. V. (2007). Characterization of complex networks: A survey of measurements. *Advances In Physics*, 56:167. (Cited on pages 73 and 79.)

Cunningham, S. J., Bainbridge, D., and Falconer, A. (2006). 'More of an Art than a Science': Supporting the Creation of Playlists and Mixes. In *International Conference on Music Information Retrieval (ISMIR)*. (Cited on pages 38, 39, 47, 52 and 93.)

de Mooij, A. (1997). Learning preferences for music playlists. Master's thesis, Technische Universiteit Eindhoven, Department of Mathmatics and Computer Science. (Cited on pages 37 and 38.)

Downie, J. S. (2006). The Music Information Retrieval Evaluation eXchange (MIREX). *D-Lib Magazine*. (Cited on pages 31 and 75.)

Downie, J. S. (2008). The music information retrieval evaluation exchange (2005–2007): A window into music information retrieval research. *Acoustical Science and Technology*, 29(4):247–255. (Cited on pages 31 and 75.)

Dreyfus, S. E. (1969). An appraisal of some shortest-path algorithms. *Operations Research*, 17(3):395–412. (Cited on page 58.)

Dunlop, M. and Crossan, A. (2000). Predictive text entry methods for mobile phones. In *Personal and Ubiqutous Computing*, volume 4, pages 134–143. Springe London. (Cited on page 23.)

Eck, D., Lamere, P., Bertin-Mahieux, T., and Green, S. (2007). Automatic generation of social tags for music recommendation. In *Neural Information Processing Systems Conference (NIPS)*. (Cited on pages 65 and 125.)

Edison, T. A. (1878). Improvement in phonograph or speaking machines. US Patent Number 200521. (Cited on page 26.)

Edison, T. A. (1891). Means for transmitting signals electronically. US Patent Number 465971. (Cited on page 26.)

Elias, P., Feinstein, A., and Shannon, C. (Dec 1956). A note on the maximum flow through a network. *IEEE Transactions on Information Theory*, 2(4):117–119. (Cited on page 73.)

Ellingham, M., editor (2007). *The Rough Guide book of Playlists*. Rough Guides Ltd., 2nd edition. (Cited on pages 24 and 101.)

Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine. (Cited on page 101.)

Fields, B., Jacobson, K., Casey, M., and Sandler, M. (2008a). Do you sound like your friends? exploring artist similarity via artist social network relationships and audio signal processing. In *International Computer Music Conference (ICMC)*. (Cited on page 84.)

Fields, B., Jacobson, K., Rhodes, C., and Casey, M. (2008b). Social playlists and bottleneck measurements : Exploiting musician social graphs using content-based dissimilarity and pairwise maximum flow values. In *International Conference on Music Information Retrieval (ISMIR)*. (Cited on pages 84, 123 and 158.)

Fields, B. and Lamere, P. (2010). Finding a path through the juke box – the playlist tutorial. Tutorial presentation at the Conference of the International Society of Music Information Retrieval (ISMIR). (Cited on page 65.)

Flexer, A., Schnitzer, D., Gasser, M., and Widmer, G. (2008). Playlist generation using start and end songs. In *International Conference on Music Information Retrieval (ISMIR)*. (Cited on pages 11, 60, 61, 65, 95, 100, 102, 123 and 158.)

Freire, A. M. (2008). Remediating radio: Audio streaming, music recommendation and the discourse of radioness. *The Radio Journal: International Studies in Broadcast and Audio Media*, 5(23):97–112. (Cited on page 27.)

Gleiser, P. and Danon, L. (2003). Community structure in jazz. *Advances in Complex Systems*, 6(4):565–573. (Cited on page 74.)

Goldberg, A. V. and Tarjan, R. E. (1988). A new approach to the maximum-flow problem. *Journal of the ACM*, 35(4):921–940. (Cited on page 73.)

Goldberg, D., Nichols, D., Oki, B. M., and Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70. (Cited on page 49.)

Graham, R. and Hell, P. (1985). On the history of the minimum spanning tree problem. *IEEE Annals of the History of Computing*, 7:43–57. (Cited on page 58.)

Grauman, K. and Darrell, T. (2004). Fast contour matching using approximate earth mover's distance. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. (Cited on page 128.)

Griffin, G., Kim, Y., and Turnbull, D. (2010). Beat-Sync-Mash-Coder: A Web Application for Real-Time Creation of Beat-Synchronous Music Mashups. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. (Cited on page 159.)

Grosche, P., Mueller, M., and Kurth, F. (2010). Cyclic Tempogram – A Mid-Level Tempo Representation for Music Signals. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Dallas, TX, USA. (Cited on page 158.)

Gruzd, A. A., Downie, J. S., Jones, M. C., and Lee, J. H. (2007). Evalutron 6000: collecting music relevance judgments. In *ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, JCDL '07, pages 507–507, New York, NY, USA. ACM. (Cited on page 34.)

Hargreaves, D. J. and North, A. C. (1999). The functions of music in everyday life: Redefining the social in music psychology. *Psychology of Music*, 27(1):71–83. (Cited on page 29.)

Hart, P., Nilsson, N., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100 –107. (Cited on page 139.)

Harwood, E. D. (2004). Staying afloat in the internet stream: How to keep web radio from drowning in digital copyright royalties. *Federal Communications Law Journal*. (Cited on page 50.)

Hayes, C. and Cunningham, P. (2000). Smart radio: Building music radio on the fly. In *Expert Systems*, pages 2–6. ACM Press. (Cited on pages 54 and 122.)

Hayes, C. and Cunningham, P. (2004). Context boosting collaborative recommendations. *Knowledge-Based Systems*, 17(2-4):131–138. AI 2003, the Twenty-third SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence. (Cited on pages 23 and 56.)

Herlocker, J. L., Konstan, J. A., Terveen, L. G., and Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. *ACM Transactions on Infinite Systems*, 22(1):5–53. (Cited on page 36.)

Herrera, P., Resa, Z., and Sordo, M. (2010). Rocking around the clock eight days a week: an exploration of temporal patterns of music listening. In *Workshop on Music Recommendation and Discovery (WOMRAD), Co-located with ACM Recommender Systems (RecSys)*, Barcelona, Spain. (Cited on pages 42, 52 and 141.)

Hockman, J. A., Bello, J. P., Davies, M. E. P., and Plumbley, M. D. (2008). Automated rhythmic transformation of musical audio. In *International Conference on Digital Audio Effects (DAFx)*, Espoo, Finland. (Cited on page 160.)

Hoffman, M. D., Blei, D. M., and Cook, P. R. (2009). Easy as CBA: a simple probabilistic model for tagging music. In *Conference of the International Society of Music Information Retrieval (ISMIR)*. (Cited on page 125.)

Hornby, N. (1995). *High Fidelity*. Indigo. (Cited on page 27.)

Howarth, P. and Rüger, S. (2005). Fractional distance measures for content-based image retrieval. In *Advances in Information Retrieval*, volume 3408 of *Lecture Notes in Computer Science*, pages 447–456. Springer. (Cited on page 128.)

Ince, R. A., Petersen, R. S., Swan, D. C., and Panzeri, S. (2009). Python for Information Theoretic Analysis of Neural Data. *Front Neuroinformatics*, 3:4–4. (Cited on pages 75 and 86.)

ISO/IEC 11172-3 (1993). *Information technology – Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s – Part 3: Audio*. ISO, Geneva, Switzerland. (Cited on page 27.)

ISO/IEC 13818-3 (1998). *Information technology – Generic coding of moving pictures and associated audio information – Part 3: Audio*. ISO, Geneva, Switzerland. (Cited on page 27.)

Jacobson, K., Fields, B., and Sandler, M. (2008). Using audio analysis and network structure to identify communities in on-line social networks of artists. In *International Conference on Music Information Retrieval (ISMIR)*. (Cited on page 79.)

Jacobson, K. and Sandler, M. (2008). Musically meaningful or just noise, an analysis of on-line artist networks. In *International Symposium on Computer Music Modeling and Retrieval (CMMR)*, pages 306–314. (Cited on pages 78 and 79.)

Jeffs, R. (1999). Evolution of the DJ Mixer Crossfader. Technical report, Rane Corporation. (Cited on page 162.)

Jehan, T. (2004a). Event-synchronous music analysis/synthesis. In *International Conference on Digital Audio Effects (DAFx)*, Naples, Italy. (Cited on page 159.)

Jehan, T. (2004b). Perceptual segment clustering for music description and time-axis redundancy cancellation. In *International Conference on Music Information Retrieval (ISMIR)*. (Cited on page 159.)

Jehan, T. (2005). *Creating Music by Listening*. PhD thesis, MIT. (Cited on page 159.)

Jiang, D.-N. J., Lu, L., Zhang, H.-J., Tao, J.-H., and Cai, L.-H. (2002). Music type classification by spectral contrast feature. In *IEEE International Conference on Multimedia and Expo (ICME)*. (Cited on page 32.)

Jolliffe, I. (2005). Principal component analysis. In *Encyclopedia of Statistics in Behavioral Science*. John Wiley & Sons. (Cited on page 59.)

Knees, P., Pohle, T., Schedl, M., and Widmer, G. (2006). Combining audio-based similarity with web-based data to accelerate automatic music playlist generation. In *ACM International Workshop on Multimedia Information Retrieval*, pages 147 – 154. (Cited on pages 58, 65, 73, 100, 102, 110, 123 and 158.)

Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480. (Cited on page 58.)

Kremp, P.-A. (2010). Innovation and selection: Symphony orchestras and the construction of the musical canon in the united states (1879–1959). *Social Forces*, 88(3):1051–1082. (Cited on page 26.)

Krumhansl, C. L. (1995). Music psychology and music theory: Problems and prospects. *Music Theory Spectrum*, 17(1):53–80. (Cited on page 29.)

Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *Annals of Mathmatical Statistics*, 22(1):79–86. (Cited on page 30.)

Kwak, H., Han, S., Ahn, Y.-Y., Moon, S., and Jeong, H. (2006). Impact of snowball sampling ratios on network characteristics estimation: A case study of cyworld. Technical Report CS/TR-2006-262, KAIST. (Cited on page 77.)

Lambiotte, R. and Ausloos, M. (2006). On the genre-fication of music: a percolation approach (long version). *The European Physical Journal B*, 50:183. (Cited on pages 74 and 90.)

Lamere, P. (2008). Social tagging and music information retrieval. *Journal of New Music Research*, 37(2). (Cited on page 125.)

Lamere, P. and Eck, D. (2007). Using 3d visualizations to explore and discover music. In *International Conference on Music Information Retrieval (ISMIR)*. (Cited on pages 49, 93 and 95.)

Langville, A., Meyer, C., and Fernández, P. (2008). Google's pagerank and beyond: The science of search engine rankings. *The Mathematical Intelligencer*, 30:68–69. 10.1007/BF02985759. (Cited on page 109.)

Lanza, J. (2004). *Elevator Music: A Surreal History of Muzak, Easy-listening, and Other Moodsong*. University of Michigan Press, 2nd edition. (Cited on page 25.)

Laplante, A. (2010). The Role People Play in Adolescents' Music Information Acquisition. In *Workshop on Music Recommendation and Discovery (WOMRAD2010), Co-located with ACM Recommender Systems (RecSys)*. (Cited on page 69.)

Law, E. and Ahn, L. (2009). Input-agreement: A new mechanism for collecting data using human computation games. In *International Conference on Human Factors in Computing Systems (CHI)*. (Cited on page 33.)

Lee, S. H., Kim, P.-J., and Jeong, H. (2006). Statistical properties of sampled networks. *Physical Review E*, 73:102–109. (Cited on page 77.)

Leong, T. W., Vetere, F., and Howard, S. (2005). The serendipity shuffle. In *The Australia Conference on Computer-Human Interaction (OZCHI)*, pages 1–4, Narrabundah, Australia, Australia. Computer-Human Interaction Special Interest Group (CHISIG) of Australia. (Cited on page 41.)

Leong, T. W., Vetere, F., and Howard, S. (2006). Randomness as a resource for design. In *The Conference on Designing Interactive systems (DIS)*, pages 132–139, New York, NY, USA. ACM. (Cited on pages 41 and 42.)

Levinthal, D. A. (1998). The slow pace of rapid technological change: Gradualism and punctuation in technological change. *Industrial and Corporate Change*, 7(2):217–247. (Cited on page 26.)

Levy, M. and Sandler, M. (2008). Learning latent semantic models for music from social tags. *Journal of New Music Research*, 37(2):137 – 150. (Cited on page 126.)

Lillie, A. (2008). Musicbox: Mapping and visualizing music collections. Master's thesis, Massachusetts Institute of Technology, MA, USA. (Cited on page 49.)

Lin, J. (1991). Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145 –151. (Cited on page 30.)

Liu, K. and Reimer, Roger, A. (2008). Social playlist: enabling touch points and enriching ongoing relationships through collaborative mobile music listening. In *The International Conference on Human Computer Interaction with Mobile Devices and Services (MobileHCI)*, pages 403–406, New York, NY, USA. ACM. (Cited on pages 54 and 122.)

Logan, B. (2000). Mel frequency cepstral coefficients for music modeling. In *International Symposium on Music Information Retrieval (ISMIR)*. (Cited on pages 29 and 74.)

Logan, B. (2002). Content-based playlist generation: Exploratory experiments. In *International Conference on Music Information Retrieval (ISMIR)*. (Cited on pages 55, 58, 65, 100 and 102.)

Logan, B. and Salomon, A. (2001). A music similarity function based on signal analysis. *IEEE International Conference on Multimedia and Expo (ICME)*, pages 745–748. (Cited on pages 29 and 74.)

Lynskey, D. (2008). *Book of Playlists; The Best of the Guardian's 'Readers Recommended'*. Aurum Press in association with Guardian Books. (Cited on pages 24 and 101.)

Maillet, F., Eck, D., Desjardins, G., and Lamere, P. (2009). Steerable Playlist Generation by Learning Song Similarity from Radio Station Playlists. In *Conference of the International Society of Music Information Retrieval (ISMIR)*. (Cited on pages 123 and 158.)

Marconi, G. (1897). Improvements in transmitting electrical impulses and signals, and in apparatus therefor. British Patent Number 12039. (Cited on page 26.)

McFee, B. and Lanckriet, G. (2009). Heterogeneous Embedding for Subjective Artist Similarity. In *Conference of the International Society of Music Information Retrieval (ISMIR)*. (Cited on page 158.)

Mitchell Parry, R. and Essa, I. (2004). Feature Weighting for Segmentation. In *International Conference on Music Information Retrieval (ISMIR)*. (Cited on page 159.)

Nagamochi, H. and Ibaraki, T. (1992). Computing edge-connectivity in multigraphs and capacitated graphs. *SIAM Journal of Discrete Mathematics*, 5(1):54–66. (Cited on page 73.)

Newman, M. E. J. (2003). The structure and function of complex networks. *SIAM Review*, 45:167. (Cited on pages 72, 73 and 78.)

O'Hara, K., Lipson, M., Jansen, M., Unger, A., Jeffries, H., and Macer, P. (2004). Jukola: Democratic Music Choice in a Public Space. In *The Conference on Designing Interactive systems (DIS)*, pages 145–154, New York, NY, USA. ACM. (Cited on pages 55 and 122.)

Oliver, N. and Kreger-Stickles, L. (2006). PAPA: Physiology and Purpose-Aware Automatic Playlist Generation. In *International Conference on Music Information Retrieval (ISMIR)*. (Cited on page 158.)

Ono, N., Miyamoto, K., Kameoka, H., and Sagayama, S. (2008). A real-time equalizer of harmonic and percussive components in music signals. In *International Conference on Music Information Retrieval (ISMIR)*, Philadelphia, PA, USA. (Cited on page 32.)

O'Shaughnessy, D. (1987). *Speech Communication: Human and Machine*, chapter 4. Addison-Wesley. (Cited on page 29.)

Pampalk, E. (2006). *Computational Models of Music Similarity and their Application in Music Information Retrival*. PhD thesis, Technischen Universität Wien. (Cited on pages 29, 30, 55, 74 and 75.)

Pampalk, E., Pohle, T., and Widmer, G. (2005). Dynamic playlist generation based on skipping behavior. In *International Conference on Music Information Retrieval (ISMIR)*. (Cited on pages 60 and 124.)

Park, J., Celma, Ò., Koppenberger, M., Cano, P., and Buldu, J. M. (2007). The social network of contemporary popular musicians. *International Journal of Bifurcation and Chaos*, 17:2281–2288. (Cited on page 74.)

Pauws, S. and Eggen, B. (2002). Pats: Realization and user evaluation of an automatic playlist generator. In *International Conference on Music Information Retrieval (ISMIR)*. (Cited on pages 62 and 124.)

Perez Gonzolez, E. and Reiss, J. (2007). Automatic Mixing: Live Downmixing Stereo Panner. In *International Conference on Digital Audio Effects (DAFx)*, Bordeaux, France. (Cited on page 159.)

Platt, J. C., Burges, C. J., Swenson, S., Weare, C., and Zheng, A. (2002). Learning a gaussian process prior for automatically generating music playlists. In *Advances in Neural Information Processing Systems (NIPS)*, volume 14, pages 1425–1432. (Cited on pages 11, 56, 57, 100, 123 and 124.)

Pohle, T. and Schnitzer, D. (2007). Striving for an improved audio similarity measure. In *The Annual Music Information Retrieval eXchange (MIREX)*. (Cited on page 33.)

Pohle, T. and Schnitzer, D. (2009). Submission to Mirex 2009 Audio Similarity Task. The Annual Music Information Retrieval Evaluation eXchange (MIREX). (Cited on pages 32 and 33.)

Pohle, T., Schnitzer, D., Schedl, M., Knees, P., and Widmer, G. (2009). On Rhythm and General Music Similarity. In *Conference of the International Society of Music Information Retrieval (ISMIR)*. (Cited on pages 32, 33 and 158.)

Pohle, T., Schnitzer, D., and Seyerlehner, K. (2010). Submission to Mirex AMS Task 2010. The Annual Music Information Retrieval Evaluation eXchange (MIREX). (Cited on pages 32 and 33.)

Pons, P. and Latapy, M. (2005). Computing communities in large networks using random walks (long version). arXiv:physics/0512106v1. (Cited on pages 74, 79 and 81.)

Ragno, R., Burges, C., and Herley, C. (2005). Inferring similarity between music objects with application to playlist generation. In *ACM International*

Workshop on Multimedia Information Retrieval international workshop on Multimedia Information Retrieval. (Cited on pages 55, 123 and 158.)

Raimond, Y., Abdallah, S., Sandler, M., and Gaisson, F. (2007). The Music Ontology. In *International Conference on Music Information Retrieval (ISMIR)*. (Cited on pages 45 and 94.)

Resnick, P. and Varian, H. R. (1997). Recommender systems. *Communications of the ACM*, 40(3):56–58. (Cited on page 36.)

Rhodes, C., Crawford, T., Casey, M., and d'Inverno, M. (2011). Investigating music collections at different scales with audiodb. *Journal of New Music Research, to appear*. (Cited on page 127.)

Rubner, Y., Tomasi, C., and Guibas, L. J. (2000). The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121. (Cited on page 30.)

Salton, G. (1989). *Automatic text processing: the transformation, analysis, and retrieval of information by computer*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA. (Cited on page 140.)

Seyerlehner, K., Schedl, M., Pohle, T., and Knees, P. (2010a). Using block-level features for genre classification, tag classification and music similarity estimation. The Annual Music Information Retrieval Evaluation eXchange (MIREX). (Cited on pages 33 and 35.)

Seyerlehner, K., Schedl, M., and Widmer, G. (2010b). Fusing block-level features for music similarity estimation. In *International Conference on Digital Audio Effects (DAFx)*. (Cited on page 33.)

Silby, B. (2007). Is the DJ an Artist? Is a mixset a piece of art? Technical report, Def-Logic Productions. (Cited on page 40.)

Sordo, M., Cyril, L., and Celma, Ò. (2007). Annotating music collections: How content-based similarity helps to propagate labels. In *International Conference on Music Information Retrieval (ISMIR)*, Vienna, Austria. (Cited on page 120.)

Steuer, R., Kurths, J., Daub, C., Weise, J., and Selbig, J. (2002). The mutual information: Detecting and evaluating dependencies between variables. *Bioinformatics*, 18 Suppl.2:S231–S240. (Cited on page 75.)

Stocky, T., Faaborg, A., and Lieberman, H. (2004). A commonsense approach to predictive text entry. In *The SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 1163–1166, New York, NY, USA. ACM. (Cited on page 23.)

Taivalsaari, A., Mikkonen, T., Ingalls, D., and Palacz, K. (2008). Web browser as an application platform: the lively kernel experience. Technical report, Mountain View, CA, USA. (Cited on page 101.)

Terrell, M. J. and Reiss, J. D. (2009). Automatic Monitor Mixing for Live Musical Performance. *Journal of the Audio Engineering Society*, 57(11):927 – 936. (Cited on page 159.)

Terveen, L., McMackin, J., Amento, B., and Hill, W. (2002). Specifying preferences based on user history. In *The SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 315–322, New York, NY, USA. ACM. (Cited on page 38.)

Turnbull, D., Barrington, L., Torres, D., and Lanckriet, G. (2008). Semantic annotation and retrieval of music and sound effects. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(2):467–476. (Cited on page 33.)

Tversky, A. (1977). Features of Similarity. *Psychological Review*, 84:327–352. (Cited on page 28.)

Tzanetakis, G. (2007). *Marsyas: a case study in implementing Music Information Retrieval Systems*. Information Science Reference. (Cited on pages 74 and 75.)

Tzanetakis, G. (2010). Marsyas Submissions to Mirex 2010. The Annual Music Information Retrieval Evaluation eXchange (MIREX). (Cited on pages 34 and 35.)

Tzanetakis, G. and Cook, P. (2002). Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5). (Cited on page 34.)

Van Gulik, R. and Vignoli, F. (2005). Visual playlist generation on the artist map. In *International Conference on Music Information Retrieval (ISMIR)*. (Cited on page 48.)

Voida, A., Grinter, R. E., Ducheneaut, N., Edwards, W. K., and Newman, M. W. (2005). Listening in: practices surrounding itunes music sharing. In

The SIGCHI Conference on Human Factors in Computing Systems (CHI), pages 191–200, New York, NY, USA. ACM. (Cited on pages 38 and 47.)

Wall, T. (2007). Finding an alternative: Music programming in US college radio. *The Radio Journal: International Studies in Broadcast and Audio Media*, 5(1):35–54. (Cited on page 26.)

Ward, M. K., Goodman, J. K., and Irwin, J. R. (2006). I Want It Even Though I Do Not Like It: Preference for Familiar but Less Liked Music. *Advances in Consumer Research*, 33:266. (Cited on page 38.)

Watts, D. J. (1999). *Small Worlds: The Dynamics of Networks Between Order and Randomness*. Princeton University Press, Princeton, NJ. (Cited on page 72.)

Weber, R., Schek, H. J., and Blott, S. (1998). A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *International Conference on Very Large Databases (VLDB)*. (Cited on page 126.)

Weber, W. (2001). From miscellany to homogeneity in concert programming. *Poetics*, 29(2):125–134. (Cited on page 25.)

Wells, A. and Hakanen, E. A. (1997). *Mass Media and Society*, chapter 8. Greenwood Publishing Group. (Cited on page 52.)

West, K., Cox, S., and Lamere, P. (2006). Incorporating machine-learning into music similarity estimation. In *The ACM Workshop on Audio and Music Computing Multimedia*, pages 89–96, Santa Barbara, California, USA. (Cited on page 33.)

Appendix A

Song Transitions in Computer-Generated Program Direction

> "Flow. Machines that describe other machines, texts that absorb other texts, bodies that absorb other bodies."
> – Paul Miller, *rhythm science*, 2004

A.1 Introduction

Recent advances in music information retrieval have lead to a more nuanced understanding of pairwise song relationships, both using content [Casey et al., 2008b; Grosche et al., 2010; Pohle et al., 2009] and various types of metadata [Fields et al., 2008b; Maillet et al., 2009; McFee and Lanckriet, 2009]. Based in part on these advances, a great deal of progress has been made in the area of automatic playlist generation or more generally, automatic program direction [Bosteels et al., 2009; Flexer et al., 2008; Knees et al., 2006; Maillet et al., 2009; Oliver and Kreger-Stickles, 2006; Ragno et al., 2005]. These automatically-assembled lists of songs can provide novel means of music discovery and exploration.

However, in many styles of music if a list of songs is simply played sequentially an important element is missing from the presentation. Much of modern dance and pop music playback is accompanied by the cultural expectation of a transition from one song to the next that obfuscates the end of the current song with the beginning of next. This is commonly referred to as *smooth* or *continuous* mixing. In many forms of human facilitated playlist presentation (*e.g.* a club or radio DJ) this is the expectation both from the expert group (*i.e.* other DJs) and the listeners/audience [Brewster and Broughton, 2006]. It is the second group that is most important to scope of this paper. It is important to acknowledge and meet the expectations of presentation style for a given playlist

of music. If this is the case, in those styles where it is appropriate, we need to augment the playback method of a playlist to make it stylistically appropriate. In this paper we will present such a playback mechanism via algorithmic smooth mixing between tracks.

We will continue in Section A.2 by examining a brief review of automatic mixing and crossfading along with relevant techniques in musical event segmentation. In Section A.3 we rigorously define several types of song to song transitions in playback. This Section continues to present a method for automatic phrase-aligned transitions between songs for playback. Finally, in Section A.4 we describe an initial implementation to qualitatively validate the described method and explore various ways to improve this technique.

A.2 Existing Methods

In this section we will briefly review relevant tools for improving song transitions. We begin with an overview of musical event segmentation, followed by discussion of how to cluster those events into larger musically meaningful groups or clusters. We will complete the section discussing other automatic mixing techniques.

A.2.1 Musical Event Segmentation and Clustering

Musical event segmentation uses various signal-processing techniques to detect onsets of musical events (*e.g.* notes) [Bello et al., 2005; Bello, 2003]. Once found, these onsets can be used to determine and define base level musically-meaningful segments (*e.g.* tatums or beats) [Jehan, 2004a, 2005].

Musical event clustering takes segments as found via the method described in Section A.2.1 and organizes the musical events into larger groups or clusters. (*e.g.* phrases) [Jehan, 2004b; Mitchell Parry and Essa, 2004].

A.2.2 Automatic Mixing

A fair amount of research has looked into the automation of the mixing process, though the primary focus in much of this work has been directed at means to automate mixing of multiple single-instrument channels into front-of-house or monitor mixes. A number of approaches exist here, but generally these tend to approach the problem as one of constrained control rules [Perez Gonzolez and Reiss, 2007; Terrell and Reiss, 2009].

While these approaches are interesting, they can only be used for song-to-song transitions once some kind of temporal alignment has been achieved. In Griffin et al. [2010] this task of beat and phase alignment has been shown in the generation of *beat-synchronous music mashups*, whereby songs which humans have deemed to be suitable together are automatically distorted in time to

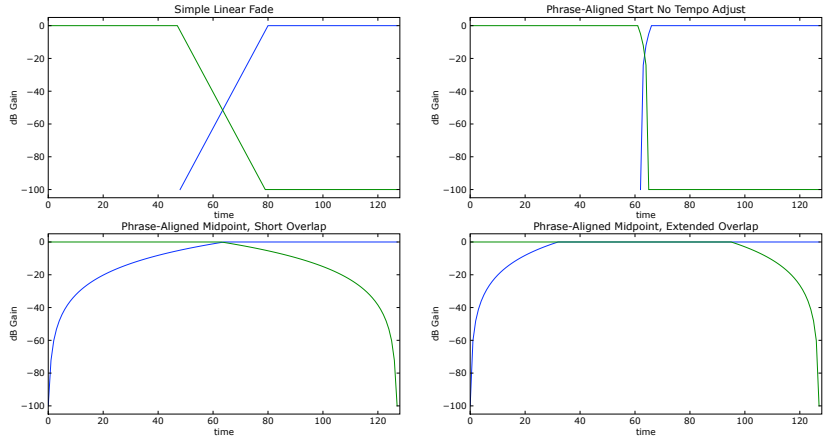


Figure A.1: Four base crossfader curves, labeled by transition type

play simultaneously in an aligned fashion. However the goal here is constant simultaneous playback. This work lacks a viable model for transitioning from one song to another.

There has also been work directed at more novel song-to-song transforms. In particular Hockman et al. [2008], which presents a means to transform one arbitrary rhythmic-pattern into another via structural rhythmic-analysis based on automatic onset and beat detection.

A.3 A Better Automated Crossfader

In this section we enumerate and define various types of song-to-song transitions. The crossfader curves for these transitions are seen in Figure A.1. We also detail an algorithmic means to execute one particular kind of transition, alluded to in Section A.1, the *smooth mix*.

A.3.1 Transition Types

For the purpose of this discussion we will be dealing with various types of song transitions in order of temporal complexity, from the simplest to the most complex. Each of these transitions moves from song a to song b.

A.3.1.1 Arbitrary-Length, Fixed-Time Crossfade

The most common form of automatic song-to-song transition, commonly known simply as an automatic crossfade. As seen in Equation A.1, the gain of either track is simply linearly increased or reduced for an arbitrary fixed time interval.

$$L_{a(t)} = \left(1 - \frac{t}{T}\right)(-G)S_a \tag{A.1}$$

where $L_{a(t)}$ is the adjusted playback gain of song a (which has an unadjusted playback gain of S_a) at time t in a fixed time crossfade of an arbitrary length T. Here t is the equivalent to time remaining in song a. $-G$ is a negative gain scaling factor equal to a magnitude power pychoacoustically equivalent to a gain factor of $-\infty$. Symmetrically, Equation A.2 will give the adjusted playback gain of the incoming song b.

$$L_{b(t)} = \frac{t}{T}(-G)S_b \tag{A.2}$$

Where $L_{b(t)}$ is the adjusted playback gain of song b and S_b is the unadjusted gain at time t in the crossfade period of length T, which for song b is equivalent to time t from the beginning of the song.

A.3.1.2 Phrase-Aligned Start No Tempo Adjust

The aim of this transition is to emulate the effect of a tempo transition in a single piece with multiple tempos. This effect is achieved by using Equations A.1 and A.2 with two adjustments. The first is that rather than have an arbitrary length of crossfade, T is set such that the first beat of the bar of song b falls exactly one beat period after the last beat of the last bar of song a. This is seen in Equation A.3.

$$T = t_{dBb} + (t_{Sa} - t_{LBa}) \tag{A.3}$$

Where t_{Sa} is the length of song a, t_{LBa} is the time offset from the start of song a where the last complete bar finishes and t_{dBb} is the time offset from the beginning of song b where the first complete bar of song b starts.

The second change is seen in type of fade in and fade out used to bring the songs out and in. In Equations A.1 and A.2 simple linear curves are employed. This will result in a decrease in the overall loudness during the transition period when both tracks have been reduced in loudness by a factor of two, resulting in a $-3dB$ decrease in the overall loudness of the mixed output. This can be avoided by introducing a overlapping log factor into the loudness curve for both tracks, for example Equation A.4.

$$L_{a(t)} = S_a \left(\log_{10} \left(\frac{10(T-t)}{T} \right) - 1 \right) \tag{A.4}$$

This curve can also be reflected and used for song a, as can be seen in Equation A.5.

$$L_{b(t)} = S_b \left(\log_{10} \left(\frac{10t}{T} \right) - 1 \right) \tag{A.5}$$

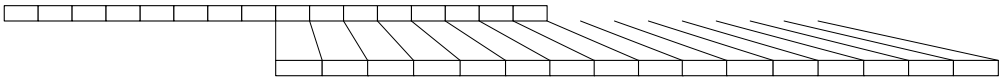


Figure A.2: A visualisation of running bar alignment, with the upper song setting the bar duration. Note that even when the song has ended the lower song will still be tempo adjusted to the estimated bar duration of the initial song.

For stylistic reasons, variations on the log curve may also be employed to modify the speed with which song a fades in or song b fades out [Jeffs, 1999].

A.3.1.3 Phrase-Aligned Midpoint, Running Bar Alignment

Adding to the last transition, the next method moves the phrase alignment point deeper into the first song by a small arbitrary number of phrases (typically one to three) with the goal of drawing out the transition period in order to further obfuscate the specific point of transition. Further, some arbitrary number of bars proceeding this phrase alignment are added to the beginning of the overlap period. This is to allow for the gradual increase in volume in song b to bring it to full ($0dB$) at the common phrase boundary point. In order for this extended overlap to avoid disrupting the established rhythmic structure of the first song, the second song's tempo must be adjusted to be identical with the first. Beyond a general tempo alignment, during the period of overlap each bar from the second song should be stretched as necessary to have the exact length of the corresponding bar from the first song. This bar alignment can be seen in Figure A.2. The bar is selected as the base in order to minimize alignment problems that may come from beats being inconstantly segmented across rhythmic structures.

One side effect of this technique is an increase in the crossfade algorithm's dependency on the accuracy identifying the underlying musical events present in the total overlap time. This is shown in Equation A.6 as the summation of durations of all bars contained during the overlap period.

$$T = \sum_{t=P-B}^{t_{LBa}} D_t \tag{A.6}$$

Where P is the aligned phrase transition in songs a and b, B is the arbitrarily-chosen number of lead in bars and D_t is the duration of the bar at time t.

The crossover curves used in this transition type are also altered slightly to best exploit the extended period of overlap. Equations A.4 and A.5 may still be used, however T is no longer to equal the entire overlap period. In Equation A.5 it should equal the duration of B, the duration of the lead in bars. In Equation

A.4 it is not necessary to rigorously define its value in this transition, though nominally it should be no greater than $\frac{t_{LBa}-P}{2}$. In both cases the remaining portion of the overlap lap period has a gain adjustment of $0dB$, leaving the loudness of the song unchanged from the original in that portion of the overlap period.

A.3.2 Maximized Relevant Overlap

The idea behind the maximized relevant overlap is to find a particular solution to the phrase-aligned midpoint, running bar alignment transition type outlined in Section A.3.1.3 that selects a number of bars in the period leading up to the phrase-aligned midpoint such that this overlap period *begins* on a phrase transition in both songs as well as having a phrase-aligned midpoint. The simplest way to do this is to search the latter half of song a and the front half of song b for phrases of the same length in bars. If one is found then that phrase becomes the leading half of the transition period between the two tracks. If no such pair of phrases is found between the two tracks there are two ways to proceed. A search can be performed for multiple sequential phrases that total the same length in bars from each track. While this can be effective, it is computational quite expensive. Alternatively, or if multiple phrases of the same length cannot be found, an arbitrary number of bars can be used as the lead in to a phrase-aligned mid point.

A.4 Discussion

We have presented a comprehensive overview of common song-transition types with a focus on automatic song to song transitions, particularly maximized relevant overlapping of songs to obfuscate the exact transition point. A brief review of relevant methods in automatic musical-event segmentation and clustering provides a practical base for implementing these ideas. This is considered in the context of an ever-improving field of automatic playlist generation and program direction, advancing the need for improved playback and presentation mechanisms.

A.4.1 Simple Implementation

The song transitions outlined in Section A.3 are not dependent on any particular musical event segmentation and clustering algorithm, though it should be noted that the more reliable the assertion of beat, bar and phrase, the closer a given method will approach the ideal. A prototype automatic mixing system which uses the echonest public api[1] to perform segmentation and clustering has been used to examine the effectiveness of the three kinds of transitions as defined in

[1] http://developer.echonest.com

Section A.3 and in particular the Maximum Relevant Overlap solution for the Phrase-Aligned Start, Phrase-Aligned Finish transition type outlined in Section A.3.2[2].

A.4.2 Further Work

From here some future work is apparent. Empirical testing of the proposed mixing algorithms would highlight possible ways to improve them. One potential testing mechanism would be to present song transitions created using the methods outlined in this paper to human listeners, alongside stylistically similar human generated song alignments and transitions. The human and machine generated mixes would be given to listeners in turn. The listeners would try to identify the song boundaries for each mix. If the methods detailed in this paper obfuscate song boundaries as well as human mixing, the error rates for manual boundary labeling in both cases would be similar. This could be considered as something of a Turing test for music mixing.

This appendix deals exclusively with time alignment and amplitude adjustment between two songs. However, most DJs make heavy use of equalization during the overlap of two songs. This equalization has many uses and automating it has the potential of improving the performance of an automatic mixing system considerably. The methods we propose in this work could be applied to specific frequency bands separately, based perhaps on content analysis, to give the effect of individual elements of the song entering or leaving the mix.

[2]source available: http://benfields.net/projects/betterfader.html