Connections in Music

Thesis

Kurt Jacobson

Centre for Digital Music
Queen Mary University of London
kurt.jacobson@elec.qmul.ac.uk

June 25, 2011

Abstract

Connections between music artists or songs provide a context and lineage for music and form the basis for recommendation, playlist generation, and general navigation of the musical universe. We examine the structure of the connections between music artists found on the web. It is shown that different methods of finding associations between artists yeild different network structures - the details of associations and how these associations are discovered impact the global structure of the artist network.

This realization informs our associations framework - based on semantic web technologies and centered around a small RDF/OWL ontology that emphasizes the provenance and transparency of association statements. We develop the MuSim Similarity Ontology and show how, combined with the concepts of linked data, it can be used to create a distributed web-scale ecosystem for music similarity.

The Similarity Ontology is evaluated against psychological models for similarity and shown to be flexible enough to accommodate each model examined. Several applications are developed based on the visualization of music artist network structures and the utilization of our associations framework along with other music-related linked data.

Acknowledgements

I would like to thank everyone in the Centre for Digital Music at QMUL. I doubt any other lab exists with such a high concentration of amazingly bright and entertaining people. In particular thanks to my supervisor Mark Sandler, who has encouraged my free-form exploration while helping me to focus my ideas.

Thanks to Chris Sutton who, in addition to reading very early drafts of this thesis, taught me loads about programming. Thanks to Matt Davies for reading drafts of this thesis as well as drafts of nearly every paper I authored while at C4DM - your patience and willingness to share your knowledge are an inspiration. Thanks to Renaud Lambiotte, and Jason Hockman for reading drafts of this thesis. Thanks to Amelié Anglade for sparking my interest in networks. Thanks to Ben Fields for your collaboration on all things MySpace. Thanks to Becky Stewart for bringing me to C4DM in the first place. Thanks to Chris Cannam for classical things and thanks to Andrew Nesbit for teaching me about smooth music.

A special thanks to Yves Raimond whose work on the application of semantic web technologies to music informatics laid the foundation for my own work. You have been a mentor and a great friend. Also thanks to Samer Abdallah, Thomas Gängler, and all the members of the Music Ontology specification mailing list for helping develop my ideas.

Thanks to Enrique Perez Gonzalez, Steve Welburn, Dan Stowell, Tim Murray Browne and Andrew Robertson for helping me play artist in C4DM Presents. Although the Presents stuff did not directly fit into this thesis the entire experience had a profound effect on me as a person and has truely expanded my horizons.

Thanks to my family: to my mother Vicki who has always been the biggest supporter of my education; to my father Ken who first sparked my interest in science; to my brother Craig who is just a great guy. Thanks to my lovely wife Larisa for all her patience and support and finally thanks to my daughter Amaya for the sunshine.

License

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

> *Disco is James Brown, hip-hop is James Brown, rap is James Brown; you know what I'm saying? You hear all the rappers, 90% of their music is me.*
>
> –James Brown, American composer and entertainer, (1933-2006)

There's a story in all music. Not just in the music, but in how it is connected - the culture and context, the history of the music. The more we know about these connections the better we can tell the stories about music. And it is these stories that make music most human. In a sense, this thesis is about making steps towards telling these stories to computers.

Increasingly we can find stories about music on the web - not just in the writings of music bloggers and reviewers, but in the data that is scattered about the web. Sometimes this music-related information is made salient with structured or semi-structured data, and sometimes it is more obfuscated. The work of Raimond [2009] has enabled a distributed web-scale information space for music-related data that provides explicit structure and precise semantics. We build on this work by extending this modeling to encompass what is perhaps the most compelling type of information related to music found on the web: information about connections - connections between music artists, connections between songs, connections between record labels, etc. These connections expound the story of music by giving us a history, a lineage, and a path to follow. And it is these connections that provide a foundation for music recommendation and playlist generation. As the title suggests, this work focuses on connections in music. Our goal is to work towards a web-scale distributed framework for modeling and re-using knowledge about musical associations. Note that throughout this work the

terms "I" and "we" are used to refer to the work of the author as supported by his advisors, University faculty, and colleagues.

1.1 Acid Brass

In 1997 British artist Jeremy Deller began a project based on a meandring lager-fueled pub conversation. The idea was to have a traditional British brass band perform some of the repetitive synth-driven acid house anthems that had become a staple of early '90s underground music. Deller's musical collaboration with the Faiery Brass Band became known as Acid Brass.[1] As a means of expounding the strange musical juxtaposition of the Acid Brass project, Deller created a sort-of mind map drawing to which he applied the grandiose titled *A History of the World*. This drawing is reproduced here in figure 1.1.

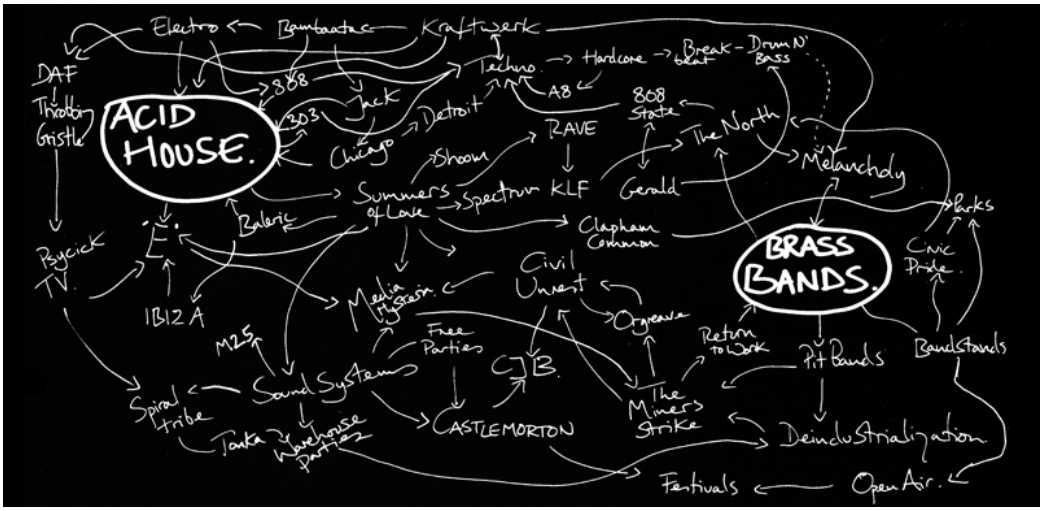


Figure 1.1: Jeremy Deller's "A History of the World" which is the basis for the Acid Brass project

The work is essentially a labeled graph of associations. Concepts including musical genre, music artists, geographical locations, electronic instruments, emotional states, and even economic systems are bound together by arrowed lines indicating some inter-concept association. Deller writes:

[1] for more information on Acid Brass see www.jeremydeller.org/ and http://www.faireyband.com/acidbrass.html

> "[*A History of the World*] stands as both the musical and social justification for the project. It's a personal interpretation but I'd like to think that we all have similar maps somewhere in our heads."

Although Deller's work makes extensive use of artistic license and some of the connections in the drawing perhaps warrant additional explanation, it shows how this simple yet powerful model - the graph model - allows us to enumerate complex associations between seemingly diparate musical styles. In our work on connections in music we make use of the graph model extensively. The graph model - containing nodes connected by edges - provides a flexible modeling framework that is easily comprehensible to both people and computers. Deller alludes to "maps somewhere in our heads". People often find graph structures an intuitive means of developing and communicating ideas, for example when creating a *mind map* diagram [Buzan and Buzan, 1996]. Furthermore, graph theory has been developing as a formal branch of mathematics since Leonhard Euler showed the *Seven Bridges of Königsberg* problem - in which a path crossing each bridge exactly once was desired - had no solution [Euler, 1741]. Today, efficient data models allow graph structures to be stored and manipulated by computers with ease [Angles and Gutierrez, 2008]. We leverage these powerful technologies to analyze music-related connections and to develop a framework for modeling connections with precise semantics.

1.2 Research Objectives

In this work, our research objectives are two-fold:

- Survey the types of music-related connections that can currently be identified on the web and examine the structure of the resulting networks of associations.
- Work towards a solution for describing music-related connections utilizing semantic web techonologies.

1.3 Requirements

Our end goal in this work is to develop a framework which embraces the diversity and complexity inherent in musical associations while providing a

computationally tractable means of leveraging these associations for music informatics tasks. The task of surveying what types of music-related connections can be discovered on the web is meant to inform the design of our associations system. Even before this survey we can specify a set of requirements for our framework for associations:

- **Heterogeneous** - A wide variety of similarities and associations related to music can already be found on the web. Our framework must be capable of expressing these connections and extensible enough to accommodate new types of connections.
- **Expressive** - Similarity and associations are often complex and multifaceted. Any pair of entities potentially have an infinite number of distinct associations between them. Our framework must accommodate the compound associations between entities and be capable of expressing the details of these associations.
- **Explicit** - The meaning of our associations should be explicit or understandable to both human users and computers and have precise semantics.
- **Auditable** - Some connections are grounded in indisputable facts while other connections are grounded in opinion while still others are the result of some algorithmic recommendation process. In our framework we must be able know *who* made a given association and *why*.
- **Distributed** - Our framework is intended to be an extension of the web and as such it must allow for a distributed information space that joins multiple data sources hosted in multiple places.
- **Queryable** - We must be able to ask questions about associations in our framework and compute answers in a reasonable amount of time.

1.4 Organization of this Work

In an effort to develop our web-scale framework for modeling musical associations in a distributed, extensible and explicit manner, this thesis is organized as follows. In chapter 2 we review some of the formalities of the graph model and some of the complex network metrics used in this work. In chapter 3 we examine a variety of music artist networks on the web. Focusing on this one specific type of musical association - the artist-to-artist connection - we

get a sense of the diversity in structure and function of musical associations that can already be found in semi-structured data on the web. In chapter 4 we review semantic web technologies and how they have been applied to the domain of music. We leverage the power of these semantic web technologies in chapter 5 in developing an association modeling framework centered around the Similarity Ontology. In chapter 6 we evaluate our association modeling framework by applying it to various cognitive models of similarity and to the concrete task of modeling a wide variety of music artist networks. In chapter 7 we describe several applications developed as part of this work and in chapter 8 we provide some conclusions and directions for future work.

1.5 Contributions of this Work

The contributions of the present work can be summarized as follows:

- A formal analysis of various types of structured music artist networks available on the web (see chapter 3)
- A framework for describing multifaceted similarity and association information (see chapter 5)
- An implementation of this association framework as a web ontology in the OWL Web Ontology Language published on the semantic web as the MuSim Similarity Ontology (see chapter 5)
- A method for evaluating a framework for associations based on models of similarity judgment from cognitive psychology (see chapter 6)
- A novel method for visualizing large labeled networks called K-Pie visualization (see section 7.2)

1.6 Published Works

- Jacobson, K, Raimond, Y, Fazekas, G, Smethurst, M. 2009. Share and Share alike, You can say anything about music on the web of data. Tutorial for the 10th Conference of the International Society of Music Information Retrieval.
- Jacobson, K, Sandler M. 2009. Interacting With Linked Data About Music. International Conference on Web Science.

- Jacobson, K, Raimond Y, Sandler M. 2009. An Ecosystem for Transparent Music Similarity in an Open World. 10th Conference of the International Society of Music Information Retrieval.
- Jacobson, K, Raimond Y, Fazekas G. 2009. You can say anything about music on the web of linked data. Tutorial for 10th Conference of the International Society of Music Information Retrieval.
- Jacobson, K, Davies M, Sandler M. 2008. The Effects of Lossy Audio Encoding on Onset Detection Tasks. 125th AES Convention. Abstract
- Jacobson, K, Fields B, Sandler M, Casey M. 2008. The Effects of Lossy Audio Encoding on Genre Classification Tasks. 124th AES Convention.
- Jacobson, K, Huang Z, Sandler M. 2008. An Analysis Of On-Line Music Artist Networks. NetSci 2008. :79-80.
- Fields, B, Jacobson K, Casey M, Sandler M. 2008. Do you sound like your friends? Exploring artist similarity via artist social network relationships and audio signal processing Proc. of ICMC.
- Jacobson, K, Sandler M. 2008. Musically meaningful or just noise, An analysis of on-line artist networks. Proc. of CMMR. :306-314.
- Fields, B, Jacobson K, Rhodes C, Casey M. 2008. Social Playlists and Bottleneck Measurements : Exploiting Musician Social Graphs Using Content-Based Dissimilarity and Pairwise Maximum Flow Values. International Symposium on Music Information Retrieval. :559–564.
- Jacobson, K, Fields B, Sandler M. 2008. Using Audio Analysis and Network Structure to Identify Communities in On-line Social Networks of Artists. Proc. of ISMIR. 2007
- Jacobson, K, Davies M, Sandler M. 2007. Towards Textual Annotation of Rhythmic Style in Electronic Dance Music. 123rd AES Convention.

Chapter 2

Network Analysis and Popularity Metrics

> *Networks are by their very nature the fabric of most complex systems, and nodes and links deeply infuse all strategies aimed at approaching our interlocked universe.*
>
> – Albert-László Barabási, *Linked*, 2003

As discussed in chapter 1, the graph model is of fundamental importance to our work. A graph is an abstract representation of a set of objects where some pairs of objects are connected by links. A vast variety of constructs can be modeled as graphs. For example, the Königsberg Bridge Problem mentioned in section 1.1 can be modeled as a graph where each landmass is a node and each bridge is link between nodes. Similarly, the *History of the World* painting can be represented as a graph where each concept is a node with links drawn between concepts. We can also consider the web a graph, where pages are nodes and links are, well, links between nodes.

Often the term *graph* and the term *network* are used interchangeably. In this work we use graph to refer to the abstract model and network to refer to some concrete instantiation of the graph model with non-trivial topological features. In this chapter we will discuss various methods of measuring network topology that have been developed in a body of work commonly referred to as *complex network theory* [Costa et al., 2007; Newman, 2003b; Barabási and Crandall, 2003; Watts and Strogatz, 1998]. 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, the Internet and the web. Many of these diverse networks exhibit several unifying characteristics such as small worldness, scale-free degree distributions, and community structure [Costa et al., 2007; Newman, 2003b; Albert and Barabási, 2002]. Some network properties and metrics we will apply to our analysis are described in section 2.1.

We will also discuss notions of artist popularity and the long tail model. These analysis techniques are described in section 2.2.

2.1 Network Properties

A given network G is described by a set of *nodes* or vertices V connected by a set of *edges* or links E. Each edge is defined by the pair of nodes it connects (i, j). The network can also be defined in terms of the *adjacency matrix* $G \equiv A$ where the elements of A are

$$A_{ij} = \begin{cases} 1 & \text{if nodes } i \text{ and } j \text{ are connected,} \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

Also, we can modify the adjacency matrix to create a *weighted* graph - where each edge has a weight associated with it usually corresponding to inter-node similarity - that is a higher weight value indicates a greater level of similarity between the nodes. To represent a weighted graph we simply replace 1 values in the matrix with some positive real number corresponding to the appropriate weight value.

If the edges imply directionality, $(i, j) \neq (j, i)$, the network is a *directed network*. Otherwise, it is an *undirected network* and the adjacency matrix is symmetric about the main diagonal.

The number of edges incident to the 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. In an undirected network we can simply refer to the overall degree of a node i as k_i. Degree can have important implications in a variety of contexts. Returning to the Königsberg Bridge Problem, Euler showed that because each node (landmass) in the network had an odd degree (an odd number of bridges attached), a path crossing each bridge exactly once was impossible. For such a path to exist (called a Eulerian path) the network must contain only nodes with even degrees or exactly two nodes with odd degrees [Euler, 1741].

2.1.1 Degree Distribution

The *degree distribution* $P(k)$ is the proportion of nodes that have a degree k. The shape of the degree distribution is an important metric for classifying a network - "scale-free networks" have a power-law distribution while "random networks" have a Poisson distribution [Barabási and Albert, 1999]. That is the fraction of nodes $P(k)$ in the network having degree k follows the relationship $P(k) \sim k^{-\alpha}$ where α is a constant whose value is typically (but not strictly) in the range of $2 < \alpha < 3$ [Newman, 2003b]. The scale-free degree distribution is a property common to many real-world networks. Conceptually, a scale-free distribution indicates the presence of a few very-popular *hubs* that tend to attract more links as the network evolves. These hubs are created because of *cumulative advantage* or *preferential attachment* - new nodes in the network have a tendency to form edges with nodes that already have a high degree. A scale-free distribution is most easily identified by plotting the cumulative degree distribution $P_c(k)$ (the proportion of nodes that have $k_i >= k$) on a log-log scale. A perfect scale-free distribution will appear as a straight line in the log-log plot [Barabási and Albert, 1999]. Scale-free distributions have implications for search [Adamic et al., 2001] and error and attack tolerance [Albert et al., 2000]. Although these implications are important we are more concerned with the intuitions we can draw from the presence (or absence) of hubs.

Alternative degree distributions also appear in many networks. An exponential distribution where $P(k) \sim e^{-k/\kappa}$ is also commonly reported [Cano et al., 2005a; Costa et al., 2007; Stumpf et al., 2005]. Such a distribution follows a straight line on a log-normal scale and indicates high-degree hubs are not present in the network. Exponential distributions can occur naturally or they can be artifacts of randomly sampled sub-networks [Stumpf et al., 2005] or information filtering [Mossa et al., 2002].

2.1.2 Average Shortest Path

Two nodes i and j are connected if a path exists between them following the edges in the network. The path from i to j may not be unique. The *geodesic path* d_{ij} is the shortest path distance from i to j in number of edges traversed. For the entire network, the average shortest path or mean geodesic distance is $\langle d \rangle$.

$$\langle d \rangle = \frac{1}{\frac{1}{2}n(n+1)} \sum_{i \geq j} d_{ij} \tag{2.2}$$

where d_{ij} is the geodesic distance from node i to node j and n is the total number of nodes in the network. In a "small-world network" the mean geodesic distance is small relative to the number of nodes in the network [Watts and Strogatz, 1998]. The largest geodesic distance in a network is known as the *diameter* or d_{max}.

2.1.3 Connected Components

It is not uncommon to find a network where nodes or groups of nodes exist disconnected from the rest of the network. Not all nodes are reachable by traversing the edges of the network. We refer to each disconnected island of nodes as a *connected component* of the network. It is common to find one component that is much larger (includes more nodes) than the other components of the network. This largest component is referred to as the *giant connected component*.

2.1.4 Network Models

Many attempts have been made to model how a complex network forms. The first formal model was the random graph model [Erdős and Rényi, 1959]. In this model edges between nodes are formed uniformly at random. This replicates the small-world phenomenon found in many real-world networks but not the scale-free degree distribution. Models that involve preferential attachment - where nodes "prefer" to form edges with other nodes that already have many edges - are able to replicate both the small-world and scale-free phenomenon found in real-world networks [Barabási and Crandall, 2003]. It is a common practice to compare a real-world network to an equivalent random network - one with the same number of nodes and edges. For example to determine small-worldness as the described in section 2.1.2, the values of $\langle d \rangle$ and d_{max} for a real-world network are often compared to the same values for an equivalent algorithmically-generated random network. If the values are in the same approximate range, the real-world network is generally considered small-world.

2.1.5 Transitivity

The transitivity or clustering coefficient estimates the probability that two neighboring nodes of a given node are neighbors themselves. In the terms of social networks, the friend of your friend is also likely to be your friend.

In terms of network topology, transitivity means a heightened number of triangles exist - sets of three nodes that are each connected to each other. For a given undirected unweighted network the transitivity is defined as

$$T = \frac{3N_{\triangle}}{N_3} \tag{2.3}$$

where $N_{\triangle}$ is the number of triangles in the network and N_3 is the number of connected triples. A connected triple is a set of three nodes where each node can be reached from every other node. The clustering coefficient is only defined for an undirected network because the definition of a triangle becomes ambiguous for the directed case Newman [2003b].

2.1.6 Betweenness

Betweenness is a measure of the number of geodesic paths that run through an edge or a node in a network. It serves as an alternative measure of a node's importance in the network structure. Betweenness can also be calculated for edges and edge betweenness is the foundation of many community detection algorithms [Girvan and Newman, 2001]. Although betweenness is more expensive to calculate, it provides a more global measure of importance where node degree only accounts for local network information [Freeman, 1977]. Betweenness is defined as:

$$B_u = \sum_{ij} \frac{\sigma(i, u, j)}{\sigma(i, j)} \tag{2.4}$$

where $\sigma(i, u, j)$ is the number of shortest paths between nodes i and j that pass through node or edge u, and $\sigma(i, j)$ is the total number of shortest paths between i and j. An efficient implmentation can be used to calculate betweenness in $\mathcal{O}(nm)$ time [Brandes, 2001].

2.1.7 Assortativity

Assortativity or homophilly relates to the tendency of nodes in a network to form connections with other nodes that share similar attributes. If a network exhibits assortative mixing, nodes tend to form links with nodes of the same type, for example, artists might tend to form friendships with other artists of the same genre. We can calculate an assortativity coefficient for an arbitrary attribute following [Newman, 2003a].

Let E be an $N \times N$ matrix with elements $E_{\gamma_i \gamma_j}$. For labels $\gamma_1, \gamma_2, \ldots, \gamma_N$, let $E_{\gamma_i \gamma_j}$ be the number of edges in a network that connect vertices of genre γ_i and γ_j. The normalized mixing matrix is defined as

$$e = \frac{E}{\|E\|} \tag{2.5}$$

where $\|x\|$ means the sum of all elements in the matrix x. The elements $e_{\gamma_i \gamma_j}$ measure the fraction of edges that fall between nodes of label γ_i and γ_j. The assortativity coefficient r is then defined as

$$r = \frac{\mathrm{Tr}(e) - \|e^2\|}{1 - \|e^2\|} \tag{2.6}$$

where $\mathrm{Tr}(x)$ is the trace (sum along main diagnol) of the square matrix x and x^2 is the matrix resulting from the squaring of all elements in x.

The assortativity coefficient will be nearly 0 in a randomly mixed network, 1 in a perfectly assortative network, and negative for a dissassortative network where nodes more often connect with nodes of different types.

A similar but distinct form of assortative mixing relates to scalar properties of the nodes in a network. For example, in a social network, it is common to see assortative mixing with respect to age - older people tend to befriend other older people while younger people tend to befriend other younger people [Newman, 2003a]. We can calculate an assortativity coefficient for a scalar value similar to the manner in which we calculated a coefficient for labels. We define a quantity e_{xy} which is the faction of all edges in the network that join together vertices with values x and y for the scalar of interest. We will assume discrete values for x and y making e_{xy} a matrix, but the continuous case can also be accommodated. As before e_{xy} satisfies the sum rules

$$\sum_{xy} e_{xy} = 1, \sum_{y} e_{xy} = a_x, \sum_{x} e_{xy} = b_y, \tag{2.7}$$

where a_x and b_y are the fraction of edges that start and end at vertices with values x and y respectively. For an undirected graph where $a_x = b_x$ the assortative mixing coefficient is then given by the standard Pearson correlation coefficient:

$$r = \frac{\sum_{xy} xy(e_{xy} - a_x b_y)}{\sigma_a \sigma_b} \tag{2.8}$$

where σ_a and σ_b are the standard deviations of the distributions a_x and b_y respectively. Again, the value of r lies in the range of $-1 \geq r \geq 1$, with $r = 1$ indicating perfect assortativity and $r = -1$ indicating perfect disassortativity.

Assortativity is often discussed with respect to degree. We will denote assortativity with respect to degree as r_k. It has been shown that in many social networks, nodes with higher degree values tend to form connections with other high-degree nodes or, to use more colloquial terms, popular people tend to befriend other popular people and conversely less-popular people tend to befriend other less-popular people [Newman and Park, 2003; Newman, 2003a]. Some music recommendation networks have also been found to exhibit assortativity with respect to degree and it has been shown this has dramatic effects on the accessibility of "long tail" artists in the recommendation network [Celma, 2008]. We will discuss these findings in more detail in section 3.3.2.

It is also important to know the expected statistical error on the value of r so that we can evaluate the significance of our results. Newman [2003a] applies the jackknife method to calculate the standard deviation σ_r. Regarding each of the m edges in a network as an independent measurement of the contributions to the elements of the matrix e, the standard deviation for r is given by

$$\sigma_r = \sqrt{\sum_{i=1}^{m}(r_i - r)^2} \tag{2.9}$$

where r_i is the value for r when the ith edge is removed. We will use this measure of standard deviation to show the statistical significance of assortativity calculations in chapter 3. We will call an assortativity value statistically significant if it is at least twice the standard deviation away from zero (recall zero is the expected assortativity value for a randomly mixed network).

We will use assortativity coefficients to relate artist node attributes to the network structures we find.

2.2 Popularity Analysis

Popularity is a nebulous concept that can be defined in a number of different ways. In our analysis of music artist networks we have a number of different metrics for popularity but the most ubiquitous metric available is that of *play counts* - how many times has a given music artist's tracks been played. It has been shown that play counts from different sources pertaining to the same artist can be highly inconsistent [Celma, 2008]. We attempt to side-step this issue by (1) comparing popularity distributions to network structure rather than other popularity distributions and (2) using alternative popularity metrics such as profile views, downloads, or friend counts as well.

2.2.1 Long Tail Analysis

We will apply some analysis related to the Long Tail model of popularity. The concept of the Long Tail was popularized by Anderson and Andersson [2006] where it is argued that products in low demand or that have a low sales volume can collectively make up a market share that rivals or exceeds the relatively few current bestsellers. A more rigorous mathematical definition of the model is given by Kilkki [2007]. Kilkki purposes a formula to fit long tail distributions of the form

$$F(x) = \frac{\beta}{(\frac{N_{50}}{x})^{\alpha} + 1} \tag{2.10}$$

Where $F(x)$ is the share of total volume covered by objects up to rank x, N_{50} is the number of objects that cover half of the entire volume, α is a factor that determines the form of the function, and β is the total volume. We make use of this model when considering music artist networks. In particular we repeat some of the analysis of the Last.fm music artist similarity network presented by Celma [2008] across several additional music artist networks. In this analysis the artist nodes are divided into three sections - the head, mid, and tail - using the following boundary definitions:

$$N_{head/mid} = N_{50}^{2/3}, N_{mid/tail} = N_{50}^{4/3} \tag{2.11}$$

We will refer to this grouping into head, mid, or tail categories as the *long tail location*. After applying a long tail location label to each node in the network, it is possible to use these labels for an assortativity coefficient calculation using equation (2.6). This calculation gives us a sense for how much mobility exists between the various popularity tiers. A value near zero indicates there are many connections between popular and less-popular artists. Celma argues quite compellingly that this has significant ramifications in terms of recommendation - a system with a high r_{lt} coefficient will have a bias towards recommending popular artists and in effect neglect the long tail. The results of Celma's Last.fm analysis are discussed in more detail in section 3.3.2.

2.2.2 Popularity Correlation Coefficient

In addition to using the Pearson correlation coefficient as a basis for scalar assortativity measures we will use at as the basis for another measure we will

refer to as the *popularity correlation coefficient*. For some networks, we have scalar values that indicate some notion of an artist's popularity (e.g. the number of times a given artist's tracks have been played). It is interesting to see if these values are correlated with the artist's degree in the network. We will refer to the Pearson correlation coefficient between node degree and some popularity metric as the *popularity correlation coefficient* ρ given by:

$$\rho = \frac{\sum_i^n (k_i - \langle k \rangle)(p_i - \langle p \rangle)}{\sqrt{\sum_i^n (k_i - \langle k \rangle)^2} \sqrt{\sum_i^n (p_i - \langle p \rangle)^2}} \tag{2.12}$$

where p is some popularity metric.

In combination with the assortativity with respect to the popularity metric, the popularity correlation coefficient gives us some notion of how the network structure relates to popularity. Although Pearson correlation coefficients are traditionally represented by r, we identify the popularity correlation coefficient at ρ to avoid confusion with the assortativity coefficient. Note the distinction between the assortativity coefficient with respect to a popularity metric $r_{popularity}$ and ρ might become more clear when you consider that $r_{popularity}$ is the Pearson correlation coefficient calculated across two one-dimensional arrays of popularity metrics that have a length exactly equal to the number of edges in the network and ρ is simply the Pearson correlation coefficient calculated between values of degree and some popularity metric (two one-dimensional arrays that have a length equal to the number of nodes).

2.3 Summary of Network Metrics

We have reviewed a variety of tools for measuring the structure of complex networks. The various network metrics and notations we will use in this work are summarized in table 2.1 for the convenience of the reader.

k	degree	r_x	assortativity with respect to x
n	number of nodes	σ_r	error in the assortativity measure
m	number of edges	ρ	popularity correlation coefficient
$\langle d \rangle$	average shortest path length	B	betweenness centrality
d_{max}	the diameter or largest geodesic distance	T	transitivity clustering coefficient

Table 2.1: A summary of the notation used for describing network properties.

In chapter 3 we will apply some of these metrics to analyze a series of music artist networks found on the web.

Chapter 3

Music Artist Networks on the Web

> *The difficulty seems to be, not so much that we publish unduly in view of the extent and variety of present day interests, but rather that publication has been extended far beyond our present ability to make real use of the record.*
>
> –Vannevar Bush, *As We May Think*, 1945

In this chapter we focus on a particular type of music-related entity found on the web - the *music artist*. Of course other music-related entities are described on the web - for example music tracks, record labels, albums, instruments, etc. - but we choose to focus on artists as they are the most central element to musical networks. Also, data about music artists on the web tends to be more readily available and easier to disambiguate than data about individual songs. We examine what sort of *structured* data about music artists and their relationships can be found on the web and we use the techniques detailed in chapter 2 to formalize our analysis. We discuss networks that were explicitly created by music experts for publication on the web including the Classical Music Navigator in section 3.4 and the All Music Guide in section 3.3.1. We sift through artist networks that are the result of crowd-sourcing massive amounts of user generated content such as the Discogs artist-release network in section 3.5. We examine online social networks of artists such as the MySpace artist network in section 3.6 and the Soundcloud artist network in section 3.7. Finally we examine networks that are the result of music informatics processing such as the Echo Nest artist

recommendation network in section 3.8 and the Last.fm artist recommendation network in section 3.3.2. But first we provide some history and context in sections 3.1 and 3.2.

3.1 Music on the Web

Music has been a part of the web almost since the web's inception. One of the earliest music-centric websites to gain real traction was the *Internet Underground Music Archive* or IUMA [1] which was started by Rob Lord, Jeff Patterson, and Jon Luini from the University of California, Santa Cruz in 1993 [Maurer, 1995]. Originally existing in the form of a Usenet newsgroup, IUMA quickly evolved into a website as its creators recognized the emerging power of the web. IUMA's goal was to help independent artists use the Internet and the web to connect with fans directly - circumventing the traditional record labels and giving unknown music artists increased exposure.

Perhaps more significantly, IUMA envisioned and created a new digital music distribution chain that involved no physical product - the artist's music and album art work were distributed directly to the consumer via the Internet as digital files. Artists were given an intuitive interface to create and maintain their own webpages and listeners were able comment on the artists' releases in a manner strikingly similar to that of today's most successful "Web 2.0" online music promotion websites [Cox, 1998]. IUMA was acquired by eMusic in June of 1999 and continued to operate for sometime. Unfortunately budget cuts and backlash from the entrenched recording industry forced IUMA to scale back its services and eventually close completely in 2006.

IUMA's webpages are no longer available, but its legacy lives on. In addition to pioneering the modern digital music distribution model, IUMA showed how the emerging technology of the web could be used to foster *connections* between music artists and music fans. A myriad of new music-centric websites have emerged since IUMA and the amount of music-related data on the web has exploded. Increasingly this data is being actively collected and leveraged by websites like Last.fm[2] and The Echo Nest[3] to create improved music listening experiences and to generate even more music-related data.

[1] formerly available at http://www.iuma.com archived content is available at http://web.archive.org/web/19961228135955/http://www2.iuma.com/

[2] http://last.fm

[3] http://the.echonest.com/

3.2 An Abridged History of Music Informatics

As the amount of content on the web exploded, improved methods for navigating that content became a necessity. The search engine was born and began to evolve - first using methods like inverse-document-frequency term-frequency indexing (IDF-TF) [Salton and McGill, 1983] and later incorporating increasingly advanced methods like Google's now famous PageRank algorithm which uses a network-based approach to ranking documents assuming the more relevant pages tend to be linked-to more often [Brin and Page, 1998]. Modern search engines are an advanced mix of these and other proprietary methods but are still based on the idea of crawling and indexing the web.

But these search engines focus on text. The field of *music information retrieval* (MIR) was born as it became evident that navigating music content was a significantly different information retrieval task with some unique challenges. The most obvious distinction being that music is manifested on the web as not just text, but as digitally-encoded audio signals or symbolic music representations - as multimedia. Researchers began to develop metrics for content-based music similarity. Using digital signal processing methods adapted from the rich history of speech recognition research, methods for calculating audio-based similarity enjoyed some success and became the focus of music information retrieval research [Logan, 2000; Tzanetakis and Cook, 2002a; Pampalk, 2006]. In short, the frame-based spectral features of an audio signal are somehow clustered to create a song-level representation. Then some distance function is applied to these song-level representations to generate a similarity score. We will discuss some audio-based similarity methods in more detail in section 3.6.4.

In parallel - in fact somewhat prior to the rise of MIR - the science of recommendation began to evolve. Recommender systems grew not out of interest in music explicitly but out of the desire to improve the online retail experience. As retailers moved their wares to web-based stores the limitations of the brick-and-mortar physical store no longer applied. As Chris Anderson describes in his popular book *The Long Tail*, this meant online retailers could stock a nearly infinite variety of goods - increasing the availability of specialty and niche products [Anderson and Andersson, 2006]. Although in retrospect, Anderson's prediction regarding the demise of mainstream hits seems to run contrary to some more recently observed trends [Elberse, 2008], he was correct in many respects. For one, shoppers

in the web world have become spoiled for choice and now need assistance finding products they might want to purchase. Of course this emerging trend was evident long before Anderson published his book and several practical solutions had already been developed. The collaborative filtering approach to recommendation was first suggested in the mid 1990s [Resnick et al., 1994] and continues to be the basis of many recommender systems to this day.

The intuition behind the collaborative filtering approach is that if a given user X expresses interest in items a,b, and c; and a second user Y expresses interest in items a and b; then user Y might also have and interest in item c. A Pearson correlation coefficient or some other calculation can be used to apply this intuition to large amounts of data - perhaps user ratings or customer purchases - to create a recommendation system.

As MIR researchers primarily focused on signals and recommender systems engineers focused on ratings data, some researchers began to recognize there is more to music. The cultural context of a piece of music can be extremely important and the nature of this context can be difficult if not impossible to tease out of signals or ratings. Increasingly people began to write about music and publish these writings on the web. The rise of user-generated content on the web meant there was an increasing amount of information about an increasing number of music-related topics. Researchers developed new approaches to music information retrieval by mining the text of these music-related webpages and blogs for key concepts and even matching these concepts with audio signal features [Whitman and Lawrence, 2002; Whitman et al., 2003]. With this web mining approach to music informatics we can, in a sense, determine similar music artists by algorithmically "reading" about music artists on the web - an approach that, arguably, encapsulates some of the cultural context and musical meaning not present in the audio signal itself.

These varied approaches to music informatics all generate a similar result on the web - some type of network - whether a network of songs connected by audio-based similarity measures or a network or listeners connected by collaborative filtering or a network of music artists connected by web document mining. It is through this lens - the lens of complex network analysis - that we will begin to study music on the web as it exists today.

3.3 Previous Artist Network Analysis

We will begin by reviewing previous analyses of music artist networks found on the web. In section 3.3.1 we discuss previous work related to the artist

networks associated with the Allmusic Guide. In section 3.3.2 we discuss previous work related to the last.fm artist recommendation network.

3.3.1 Allmusic Guide Artist Network

The All Music Guide website (AMG)[4] provides information about directed influence relationships. AMG focuses on contemporary popular music but also contains information about classical composers. There is a wealth of additional data in the AMG website including relational data like "similar artists" and "member of" relations connecting artists with other artists as well as data about albums, tracks, genres, and styles.

The AMG website is curated by experts - professional music critics paid to collect and edit information. AMG has been a part of the Internet for a long time - in fact predating the web as a Gopher site started in 1991 by popular-culture archivist Michael Erlewine. The AMG site is actively maintained and updated to this day. AMG licenses their proprietary database to partners for a fee, funding the maintenance of the site and paying their expert content creators. This data licensing scheme makes the AMG data less attractive in the context of this thesis. Such licensing schemes prohibit us from applying our own modeling to the data and re-publishing or re-purposing the data (at least without paying a fee). Therefore we present no new analysis of the AMG networks here.

Fortunately, several formal analyses of various views of the AMG artist network have already been performed [Park et al., 2006; Cano and Koppenberger, 2004; Cano et al., 2005a; Giaquinto et al., 2007; Celma, 2008]. All of these analyses deal with the artist similarity network and some exclusively so [Cano and Koppenberger, 2004; Cano et al., 2005a; Celma, 2008]. In the artist similarity network each node represents a music artist or music group and each edge represents a similarity relationship as prescribed by the AMG editors. Because the AMG data also includes additional relational information different networks can be constructed where, instead of using similarity as the edges, influence or collaboration relationships can be used to construct edges between artist nodes.

In [Giaquinto et al., 2007] a sub-sample of the AMG artist network is examined. This network is sampled by selecting only artists that are labeled as "contemporary jazz". The authors examine the directed influence network and find that it is sparsely connected and contains many fractured components. This is contrary to the Classical Music Navigator influence network

[4]available at http://allmusic.com/

which is made up of one connected component as we will discuss in section 3.4. This contradiction demonstrates the subjectivity of such relational data - the structure of any "musical influence" network is highly dependent on the definition of influence used to construct the network and the breadth of artist coverage. For a moment, let us assume infinite breadth of artist coverage - we have an army of tireless music critics creating our influence network using their expert knowledge and they've managed to cover every known music artist that ever was or ever will be. Depending on the criteria used to determine an influence relationship, we might find that all music artists across all musical styles, periods, and genres are somehow part of a giant connected network. Or with more conservative criteria for influence we might still find a fractured network with many smaller components. In reality we will never have infinite coverage and incomplete coverage will often result in a fractured network with many disconnected components. However, the tendency of these networks to contain hubs favors the existence of a giant connected component.

An exhaustive crawl of the artist similarity network reveals that it too is fractured but to a lesser extent. In [Park et al., 2006] the authors observe that the size of the giant connected component of the AMG artist similarity network is 94% of the entire network in terms of nodes. This fits with our intuition that as coverage increases, the proportional size of the giant connected component increases. The AMG similarity network exhibits an exponential decay in the cumulative degree distribution instead of conforming to the more common "scale-free" power-law distribution [Cano et al., 2005a]. However, it is possible that the expert editors are actually truncating what is actually a scale-free distribution by filtering links between normal artists and hub artists to achieve the desired web page characteristics. The truncating of scale-free distributions into exponential decays has been described by Mossa et al. [2002].

Park et al. [2006] also provide an analysis of the collaboration network. This is an undirected network where music artists are considered to be connected if they have worked together. The collaboration network is similar in many respects to the similarity network - it exhibits a high level of transitivity ($T = 0.182$) and there exists a giant connected component comprising 89% of the network in terms of nodes. However the collaboration network does exhibit a power-law degree distribution following the more common scale-free structure. In practical terms this means that there are music artists that act as hubs and have collaborated with many other artists - orders of magnitude more than the average. This conforms to the structures we find when investigating the Discogs artist-release network (very similar to a collaboration

network) in section 3.5. Using both node degree and betweenness centrality as measures in the AMG collaboration network, the Park et al. [2006] show that the most collaborative artists tend to be rhythm section players such as Paulinho Da Costa and Jim Keltner [Park et al., 2006].

In the AMG website we find structured data about music artists that allows us to build an artist network with at least three distinct edge types - "similar", "influence", and "member of" relations. The previous work we have discussed here shows how utilizing different edge types for artist network construction yields different network structures.

3.3.2 Last.fm Artist Network

Last.fm[5] is a website that allows users to record their listening habits using a light-weight client application. Users can also apply free-text tags to artists and tracks using a web interface. A proprietary mix of collaborative filtering over listening habits data and tag aggregation are used to create artist recommendations. These artist recommendations give rise to a music artist network where the nodes are music artists and the edges denote an inter-artist recommendation or similarity.

Several studies have examined data associated with Last.fm. Jäschke et al. [2007] focus on the tagging features and discusses tag recommendation in folksonomies. Using data about listening habits from Last.fm Lambiotte and Ausloos [2005] treats listeners and music artists as nodes in a bipartite graph and develops a percolation-based approach to community detection. Teitelbaum et al. [2008] examines the Last.fm artist similarity network looking for community structures and applying a method for role detection.

However, here we will focus on the analysis of the Last.fm artist similarity network presented by Celma [2008]. Some of the network statistics reported by Celma are presented in table 3.1.

The average geodesic distance is small relative to the size of the network ($\langle d \rangle = 5.64$) as is the diameter ($d_{max} = 10$) indicating the last.fm artist recommendation network is a small-world network. We see a high level of clustering significantly greater than that of an equivalent random network and assortative mixing with respect to genre. Celma [2008] also reports a power-law degree distribution in the last.fm artist recommendation network with $\alpha = 2.31$.

Interestingly, a very high assortativity coefficient with respect to degree

[5] see http://last.fm/

n	m	$\langle d \rangle$	d_{max}	$T(T_{random})$	r_k	r_γ
122,801	1,735,179	5.64	10	$0.230(1 \times 10^{-4})$	0.920	0.343

Table 3.1: Network statistics from the Last.fm artist network from Celma [2008] including the number of nodes n, the number of edges m, size of the giant component n_{S_0}, average shortest path length $\langle d \rangle$, diameter d_{max}, clustering coefficient for the undirected view of the network T and clustering coefficient for an equivalent random graph, the assortativity with respect to degree r_k, and the assortativity wrt to genre r_γ

($r_k = 0.92$) is reported. This means that in the Last.fm artist recommendation network artists with high degree values (artists that are recommended often) tend to connect with other high degree artists. Conversely low-degree artists tend to connect with other low-degree artists. Celma suggested this has implications for the artist recommendation process. Using an artist's play counts as a measure of popularity, artists in the Last.fm network sample were classified as either part of the *head*, *mid*, or *tail* of the popularity curve. Then an assortativity calculation was performed with respect to segment of the popularity curve (as we described in section 2.2.1. It was found that zero links existed from the head directly into the tail. Therefore the last.fm recommeder system generally fails to present unknown or new "long tail" artists to the user. Celma reports that the Last.fm artist network has an assortativity with respect to long tail location of $r_{lt} = 0.397$.

3.4 The Classical Music Navigator

Let us begin discussing our original analysis of music artists on the web with one of the earlier collections of music-related structured data to appear on the web. The Classical Music Navigator[6] (CMN) is a website that maps the influence connections between a collection of classical music composers. The CMN website was conceived by Charles H. Smith, a professor of library studies at Western Kentucky University, in 1993 as a resource for discovering classical composers and their works. The intuition was, if a listener enjoyed a particular composer x she might also enjoy composers that influenced x or that were influenced by x.

With the help of his colleagues, Smith created a directed network of classical composers where each node represents a composer and each edge rep-

[6]available at http://people.wku.edu/charles.smith/music/index2.htm

resents a directed relationship of influence. The directed network is encoded as a series of static HTML pages and published on the web. According to Smith the network was constructed using purely objective criteria including:

- number of available recordings of the composers' music as listed in several standard music recordings catalogs (Schwann, Gramophone, etc.);
- number of items on and by the composers held by participating institutions in the OCLC "WorldCat" database (the combined holdings of over 20,000 libraries);
- overall size/length of entries on the composers in about a dozen standard reference works

However, this brief description of the process seems to be the limit of the transparency of the network construction process Smith [1993]. The original network contains 444 composers but after removing entries encoded with problematically ambiguous HTML we are left with 426 composers for our network analysis. The network statistics derived from our original analysis of the Classical Music Navigator composer network are summarized in table 3.2.

n	m	$\langle d \rangle$	d_{max}	T (T_{random})	$r_k(\sigma_r)$
426	1780	3.39 (3.06)	10 (6)	0.144 (0.019)	$-0.161(5.29 \times 10^{-3})$

Table 3.2: Table of network statistics for the Classical Music Navigator composer network including the number of nodes n, the number of edges m, size of the giant component n_{S_0}, average shortest path $\langle d \rangle$, diameter d_{max}, clustering coefficient for the undirected view of the network T and clustering coefficient for an equivalent random graph, and the assortativity with respect to degree r_k.

This network is comprised of one large connected component - probably an artifact of the network construction methodology. Perhaps Smith et al. only included composers for which they could make an influence connection to their existing collection of composers. Both the average geodesic distance $\langle d \rangle$ and the maximum geodesic distance d_{max} are slightly larger than that of an equivalent random network but still small enough to consider indicative of a small-world network. The clustering coefficient T is an order of magnitude higher than that of a random network. This coupled with the relatively large diameter of the network suggests a structure with clearly defined communities, however explicit community structure analysis is left to future work. We

also find the network is slightly dissassortative with respect to degree with r_k having a negative value. This means composers have a some tendency towards forming influence connections with other composers that have a *different* degree value. The corollary to this finding is that influential composers do *not* have a tendency to influence each other.

We can also examine the network in terms of degree values. In table 3.3 we list the composers with the highest degree values in terms of overall degree k, in degree k_{in}, and out degree k_{out}. Since the edges in the CMN network correspond to an "influenced by" relation where the source is influenced by the target, we can infer that the most influential composers have the highest in-degree values of k_{in}. Not surprisingly the list of composers with the largest values of k_{in} as shown in table 3.3 roughly corresponds to the list of most influential composers generated by Smith [1993].

k	artist	k_{in}	artist	k_{out}	artist
83	Debussy, Achille-Claude	68	Wagner, Richard	18	Ravel, Maurice
82	Wagner, Richard	66	Debussy, Achille-Claude	17	Debussy, Achille-Claude
79	Stravinsky, Igor	66	Stravinsky, Igor	14	Wagner, Richard
70	Bach, Johann Sebastian	58	Bach, Johann Sebastian	14	Liszt, Franz
61	Mozart, Wolfgang Amadeus	48	Mozart, Wolfgang Amadeus	13	Mozart, Wolfgang Amadeus
55	Liszt, Franz	44	Beethoven, Ludwig van	13	Stravinsky, Igor
55	Beethoven, Ludwig van	42	Schoenberg, Arnold	13	Britten, Benjamin
54	Schoenberg, Arnold	41	Liszt, Franz	12	Reger, Max
48	Ravel, Maurice	32	Schumann, Robert Alexander	12	Ligeti, György
42	Brahms, Johannes	32	Chopin, Frédéric	12	Bach, Johann Sebastian

Table 3.3: Highest degree nodes for the CMN network by overall degree k, out degree k_{in}, and in degree k_{out}

We can also quantify a composer's importance in the network with the *betweenness* measure as described in section 2.1.6. Recall that a node with a higher betweenness value has a higher number of geodesic paths that pass through it. A listing of the composers with the highest betweenness values is given in table 3.4: We see that many of the same composers with the highest k_{in} values also appear in our list of composers with the highest betweenness values. Given that Wagner has the highest value for k_{in} and Bach has the highest betweenness, we could say that Wagner has the highest level of direct influence but Bach is more important in the propagation of influence - at least according to the structure of the CMN network. We leave it to the musicology community to judge the validity of this statement.

Finally, we examine the degree distributions for the CMN network. The cumulative degree distribution for the CMN network is shown in figure 3.8. A log-normal scale is used to plot the distribution of k_{in} which most closely approximates an exponential distribution while a log-log scale is used to show the approximation of a power-law distribution for nominal values of k_{out}.

B	artist
9540.07	Bach, Johann Sebastian
5315.73	Stravinsky, Igor
5179.48	Debussy, Achille-Claude
4978.14	Mozart, Wolfgang Amadeus
3986.95	Beethoven, Ludwig van
3686.27	Wagner, Richard
3617.96	Schoenberg, Arnold
2903.47	Handel, George Frideric
2685.47	Sweelinck, Jan Pieterszoon
2460.29	Ravel, Maurice

Table 3.4: Classical composers with the highest betweenness values

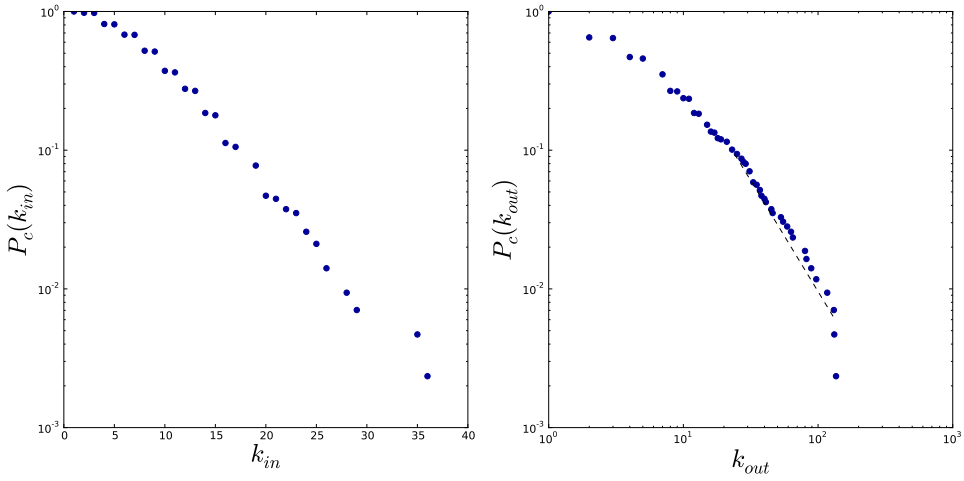


Figure 3.1: The cumulative degree distribution for the CMN composer network

3.5 Discogs Artist-Release Network

Discogs[7] is a community-built database containing information on artists, labels, and recordings. Discogs primary function is to maintain artist discographies - listings of all the releases associated with a particular artist - with a special emphasis on vinyl releases.

The Discogs artist-release network is defined in a slightly different manner

[7]available at http://discogs.com/

than the AMG collaboration network we discussed in section 3.3.1. Two artist nodes are considered to be connected by an edge if they have appeared together on an album or release. This definition of an edge is in some sense broader than that used in the AMG collaboration network. In our Discogs artist-release network an edge between artists might indicate collaboration, but it might also indicate they appeared on the same compilation release and never collaborated directly.

The Discogs artist-release network is constructed using an RDF translation of the Discogs database[8] and a SPARQL query against the RDF translation is used to find all releases that have two or more artists listed as contributors. We will discuss RDF and the SPARQL query language in depth in chapter 4 but we will list the SPARQL query here for completeness:

```
SELECT DISTINCT ?release ?artist WHERE
{
?release a mo:Record ;
  foaf:maker ?a, ?b ;
  foaf:maker ?artist .
FILTER ( ?a != ?b ) .
}
```

Listing 3.1: SPARQL query for constructing the Discogs artist-release network

This query returns any release that has two or more artists credited with its creation. The results of this query are parsed such that for each value of `release` we create a fully-connected network from the associated values of `artist`. Thus we build the Discogs artist-release network. Note that we also remove the "place holder" artists "various" and "unknown artist." We summarize some of the network statistics for the Discogs artist-release network in table 3.5.

The Discogs artist-release network is highly fragmented. The giant connected component contains only 49.4% of the nodes in the network and there are a total of 25,647 disconnected components in the network. However, the second largest component is orders of magnitude smaller than the giant component consisting of only 46 artist nodes. Of the 25,647 components, 19,156 (about 75%) contain only two artist nodes. If we sum all these two-artist components we see that 30.5% of the artists in the Discogs artist-release network have only appeared on a release with one other artist (or they have

[8] available at http://api.talis.com/stores/discogs/services/sparql

n	m	$m_{multiple}$	n_{S_0}	$\lvert S \rvert$	$\langle n_S \rangle$
125,498	161,967	476,931	62,043 (49.4%)	25,647	4.93

Table 3.5: Table of network statistics for the Discogs artist-release network where n is the number of artist nodes, m is the number of release edges, n_{S_0} is the number of nodes in the largest connected component, $|S|$ is the total number of components and $\langle n_S \rangle$ is the average number of nodes in each connected component in the network.

appeared on several different collaborative releases but always with the same artist). This is a rather stark contrast to the AMG collaboration network where the largest connected component contained 89% of the artist nodes in the network. Recall that the AMG collaboration network was hand-curated by a team of professional music critics with the purpose of allowing users to navigate from artist to artist. The Discogs website is entirely crowd-sourced with the aim of providing free and open metadata for music artists and releases. There is no explicit impetus to provide a navigable artist collaboration network. In fact, we went to considerable effort to extract this data from the Discogs website. In this sense we can assume that the Discogs coverage might be broader but the AMG coverage is certainly deeper. The available network statistics support this assumption - the Discogs artist-release network has considerably more artist nodes (125,500 in Discogs vs. 34,724 in AMG) but, when we collapse multiple edges in the Discogs artist-release network, the AMG collaboration network contains a comparable number of edges (161,967 in Discogs vs. 123,122 in AMG).

We have mentioned that the Discogs artist-release edges do not directly correspond to collaboration - they correspond to artists appearing on the same release. The release could be a collaborative efforts between the artists or the release could be a compilation where the artists are not directly collaborating at all. We can gleam some more insight by examining how many artists appear on each release. Figure 3.2 shows a cumulative distribution for the number of artists appearing on each release η: We can see that 99.4% of releases actually have 5 or fewer contributors. This suggests that the vast majority of the releases in our data set are either direct collaborations or compilations with a rather narrow focus containing multiple songs from each artist. As it turns out, there is a third scenario that is abundantly common in the Discogs artist-release network - the composer-performer-conductor scenario.

In the Discogs data the role of a particular contributor is not always

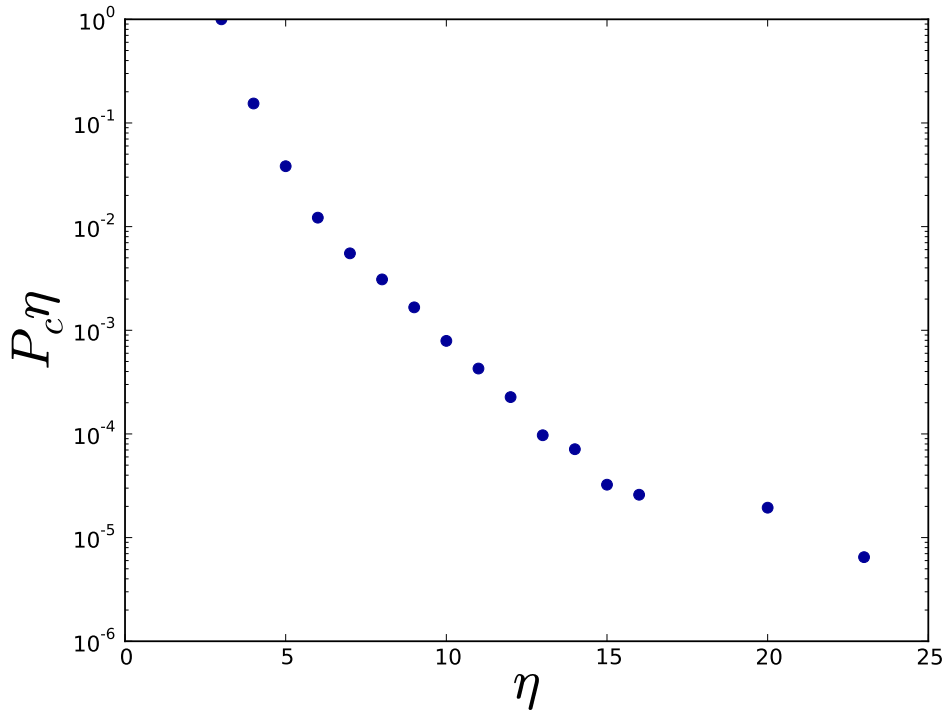


Figure 3.2: The cumulative distribution for η the number of artists appearing on each release for the Discogs artist-release network. We see that 99.4% of releases have 5 or fewer contributors.

clearly demarcated. In our modeling we assume only one type of artist-release relationship exists which is clearly not the case. In classical music, it is common to have a performer (i.e. The London Symphony Orchestra), a conductor (i.e. Sir Eugene Goossens), and a composer (i.e. Hector Berlioz). In our overly simplistic scheme these three entities are all considered as contributing artists for the release in question. If we examine the list of the highest degree artists we can infer this scenario had a big impact on our network structure. The list of artists with the highest degrees and highest betweenness values in the Discogs artist-release network is given in table 3.6. We can see that classical composers dominate the highest-degree list. This is of course an artifact of the composer-conductor-performer scenario. Filtering out this artifact is left to future work. We see some more variation in the high-betweenness list which includes pop legends like David Bowie and Michael Jackson, hip-hop stars like Jay-Z and Missy Elliott, and even the avantgarde noise artist Merzbow. Intuitively one would expect genre-spanning artists

k	artist	$B \times 10^7$	artist
661	Johann Sebastian Bach	12.0	David Bowie
654	Ludwig van Beethoven	9.1	Merzbow
638	Wolfgang Amadeus Mozart	8.4	Jay-Z
353	Piotr Illitch Tchaikovsky	8.3	Missy Elliott
340	Johannes Brahms	7.3	Michael Jackson
301	Franz Schubert	7.0	Elephant Man
275	The London Symphony Orchestra	6.0	Thurston Moore
273	Antonio Vivaldi	5.9	RedSK
260	Georg Friedrich Händel	5.8	The London Symphony Orchestra
250	Maurice Ravel	5.8	Beck

Table 3.6: Artists with highest degree k and betweenness B values in the Discogs artist-release network after removal of multiple edges

to have the highest betweenness values in the Discogs artist-release network - connecting clusters of otherwise genre-specific co-releasing artist. This is perhaps what we see with the likes of David Bowie, Jay-Z, and Beck. Also, consider that inter-generational artists with a long career would potentially have a higher betweenness (perhaps David Bowie and Michael Jackson fit this description). However, we also see more obscure but highly collaborative artists that serve as a conduit connecting clusters of artists specializing in smaller niche genres to the giant connected component of collaborative artists. This is what we see in Merzbow - who collaborates mostly with Japanese musicians and obscure electronica artists - and RedSK - a genre-hopping electronica serial-collaborator who as of this writing has fewer than 20,000 plays[9] on Last.fm.

If we examine some of the network statistics for the giant connected component of the Discogs artist-release network we see some familiar patterns. The network statistics for the giant connected component are summarized in table 3.7. We see a diameter d_{max} and an average geodesic distance $\langle d \rangle$ that can be considered in the small-world range. The clustering coefficient is orders of magnitude higher than that of an equivalent random graph. Notably we see the assortativity coefficient r_k is slightly positive but close to zero indicating a network where highly collaborative artists have no strong tendency to preferentially collaborate with other highly collaborative artists.

In figure 3.3 we plot the cumulative degree distribution of the giant connected component. In the log-log plot we can see the distribution approxi-

[9]see http://www.last.fm/music/RedSK

n	m	$\langle d \rangle$	d_{max}	T	$r_k(\sigma_r)$
62,043	118,290	9.04 (8.35)	31 (17)	0.110 4.0×10^{-5}	$0.066(2.4 \times 10^{-3})$

Table 3.7: Network statistics for the giant connected component of the Discogs artist-release network where n is the number of nodes, m is the number of edges, $\langle d \rangle$ is the average geodesic distance, d_{max} is the diameter, T is the clustering coefficient, r_k is the assortativity coefficient with respect to degree, and σ_r is the error in the assortativity coefficient

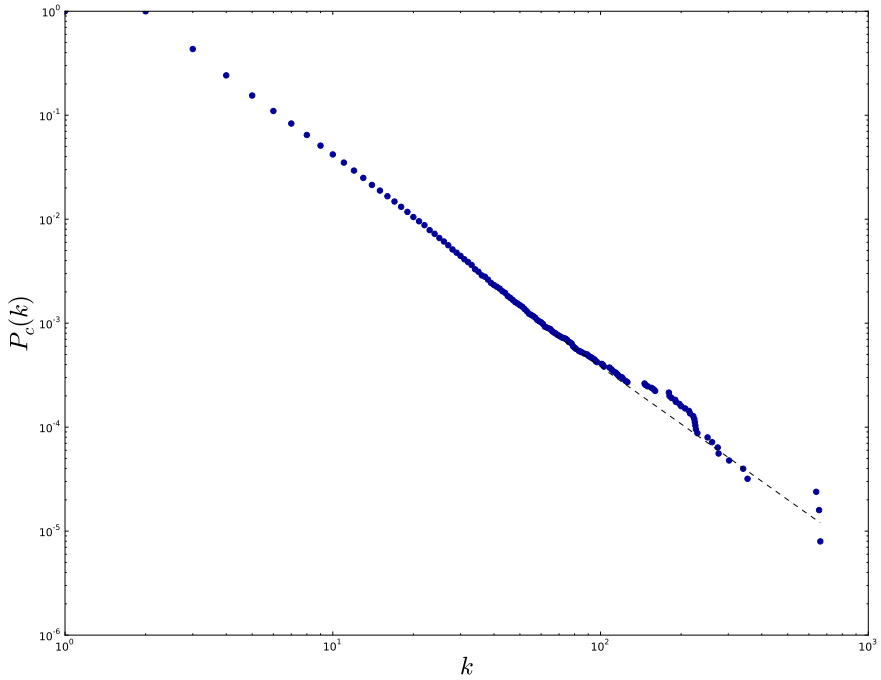


Figure 3.3: The cumulative degree distribution for the Discogs artist-release network and the power-law fit approximation where $\alpha = 2.83$ plotted on a log-log scale

mates a straight line indicating a power-law distribution with an exponent of $\alpha = 2.83$. For degrees above $k = 400$ we see some deviance from the power-law fit. Note this plot is for the network after collapsing multiple edges into single edges. If we keep multiple edges we see a very similar distribution.

In the Discogs artist-release network we see a network that is largely fragmented with one giant component and many very small components.

The largest connected component conforms to the small-world and scale-free structures found in most real-world networks. We now turn our attention to another crowd-sourced artist network - the MySpace artist network.

3.6 MySpace Artist Network

MySpace[10] is a social networking website that allows users to maintain profiles of personal information and specify friendship relationships. Although the site's popularity has waned somewhat recently, it continues to provide a standard format for music artist promotion. Although exact figures are not made public, recent blog chatter suggests there are over 7 million artist pages[11] on MySpace.

Artists ranging from bedroom electronica amateurs to multi-platinum mega-stars publish MySpace pages. These MySpace artist pages typically include some media - usually streaming audio, video, or both - and a list of "friends" specifying social connections. This combination of media and a user-specified social network provides a unique data set that is unprecedented in both scope and scale.

3.6.1 Sampling MySpace

The MySpace social network presents a variety of challenges. For one, the massive size prohibits analyzing the graph in its entirety, even when considering only the artist pages. Therefore we sample a small yet sufficiently large portion of the network as described in section 3.6.2. Also, 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. Our data is collected using web crawling and screen scraping techniques.

Artist Pages

Again, 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

[10] available at http://www.myspace.com/

[11] http://scottelkin.com/archive/2007/05/11/Myspace-Statistics.aspx ~25 million songs, ~3.5 songs/artist, ~7 million artists - last accessed 26/11/2008

audio player application. We use the presence or absence of this player to determine whether or not a given page is an artist page.

A MySpace page will most often include a top friends list. This is a hyperlinked list of other MySpace accounts explicitly specified by the user as friends to highlight on the user's page. 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. This reduces the amount of noise in the sampled network but also artificially limits the out degree of each node. Our sampling method is based on the assumption that artists specified as top friends have some meaningful musical connection for the user - whether through collaboration, stylistic similarity, friendship, or artistic influence.

Each MySpace artist page includes between zero and three genre tags. The artist selects from a list of 119 genres specified by MySpace. These genre label associations were also collected. In our sample set, around 2.6% of artists specified no genre tags. We also collect the country, number of profile views, and total number of friends associated with each MySpace artist in the sample[12]

The audio files associated with each artist page in the sampled network are also collected for feature extraction. Cached versions of the audio files are downloaded and audio features are extracted.

3.6.2 Snowball Sampling

For the MySpace artist network, snowball sampling is the most appropriate method [Ahn et al., 2007]. Alternative methods such as random edge sampling and random node sampling would result in many small disconnected components and not provide any insight to the structure of the entire network [Lee et al., 2006]. In snowball sampling, a first seed node (artist page) is included in the sample. Then the seed node's neighbors (top friends) are included in the sample. Then the neighbors' neighbors. This breadth-first sampling is continued until a particular sampling ratio is achieved. We ran-

[12]this data was collected approximately 8 months after the original sample, ideally it would have been collected at the same time

domly select one seed node[13] and perform 6 levels of sampling - such that in an undirected view of the network, no artist can have a geodesic distance greater than 6 with respect to the seed artist - 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 in [Kwak et al., 2006].

With snowball sampling there is a tendency to over-sample hubs because they have many links and are easily picked up early in the breadth-first sampling. This property would reduce the degree distribution exponent and produce a heavier tail but preserve the power-law nature of the network [Lee et al., 2006].

3.6.3 Network Topology

Our MySpace artist network samples exhibit many of the network characteristics common to social networks and other real-world networks [Newman, 2003b]. Some of the network topological metrics are summarized in table 3.8.

	n	m	$\langle d \rangle$	d_{max}	T
directed	15,478	120,487	6.43 (4.93)	16 (10)	-
undirected	15,478	91,362	4.48 (4.18)	9 (7)	0.219 (7.1×10^{-4})

Table 3.8: 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, $\langle d \rangle$ is the mean geodesic distance, d_{max} is the diameter, and T is the clustering coefficient. The clustering coefficient is undefined for directed networks

Given our sampling method dictates that only top friend relationships are included, the network samples are *directed* and each edge implies directionality. This is because the top-friend relationship may not be reciprocated – node j might be a top friend of i but i might not be a top-friend of j. If we define edges in terms of the nodes they connect this means $(i, j) \neq (j, i)$. Recall that in a directed graph each node i will have an *in-degree* k_i^{in} - indicating the number of edges terminating at node i – and an *out-degree* k_i^{out} - indicating the number of edges that originate at node i.

[13] our randomly selected artist is French rapper Karna Zoo http://www.myspace.com/karnazoo

To simplify analysis we can convert to an *undirected* network. Each edge is then considered bi-directional, that is $(i, j) = (j, i)$, and if a reflexive pair of edges existed in the directed graph, only one bi-directional edge exists in the undirected graph. In our primary sample, when converting to an undirected network we see the number of edges decrease by about 25%. This means that nearly half of all top friend relationships are reciprocated in our primary network sample and slightly less for the auxiliary sample.

The value of the clustering coefficient $T = 0.219$ indicates a high level community structure - an equivalent random network would have a transitivity of $T_{rand} = \frac{\langle k \rangle}{n} = 7.1 \times 10^{-4}$. It should be noted that it is impossible to say whether or not this value is indicative for the entire MySpace artist network because of our limited sample size [Kwak et al., 2006].

The cumulative degree distributions for the network sample are plotted in figure 3.4. Both the in-degree and out-degree distributions are plotted. Notice the in-degree distribution is plotted on a log-log scale. For moderate degree values ($35 < k_{in} < 200$), the in-degree distribution follows a power-law decay with an exponent of $\alpha = 3.39$ indicated by the fit of a straight line in the log-log scale. The power-law fit breaks down for high and low values of k_{in}. Due to finite size, the power-law is expected to break down for high values of k_{in}. Similar "multi-scale" degree distributions have been reported for citation networks and movie actor networks [Amaral et al., 2000] as well as in online social networks [Ahn et al., 2007].

The cumulative out-degree distribution is plotted on a linear-log scale. For moderate values of follows of k_{out} we see an exponential decay indicated by the fit of a straight line in the linear-log scale. Such distributions have been reported in a variety of networks including some music artist networks such as the AMG artist similarity network [Cano et al., 2005a]. Recall that there is an out-degree limit imposed by MySpace which restricts the maximum number of top friends ($k_{out} \leq 40$). This type of information filtering could actually be truncating a power-law decay [Mossa et al., 2002].

The cumulative combined degree ($k = k_{in} + k_{out}$) distribution exhibits a multi-scaling behavior very similar to the in-degree distribution – following a power-law for moderate values of k.

From the in-degree distribution, we can see that there exists a few very well-connected hub artists with an in-degree much higher than the average. The artist with the most in-links ($k_{in} = 222$) is Grammy-award-winning rapper *T.I.*[14] immediately followed by Grammy-award-winning producer *Tim-*

[14] http://www.myspace.com/trapmuzik

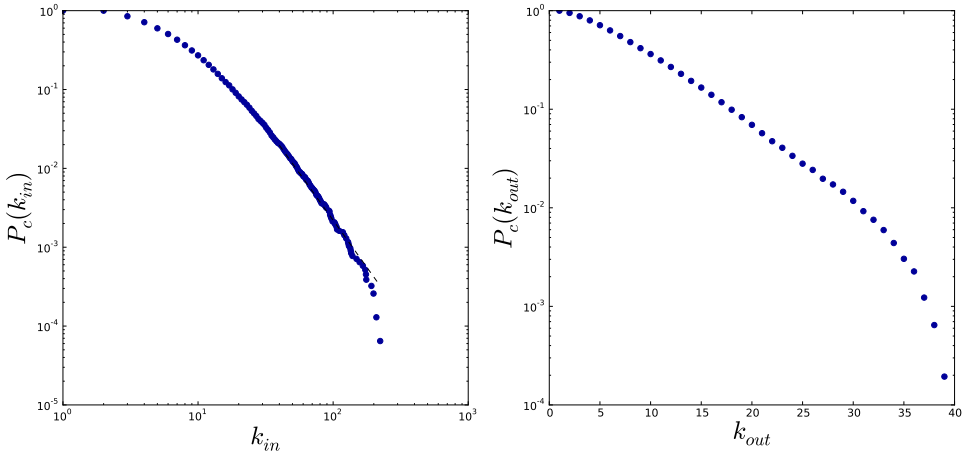


Figure 3.4: The cumulative degree distributions for (a) the in degree plotted on a log-log scale and (b) the out degree plotted on a log-linear scale in the MySpace artist network

baland.[15] The 10 artists with the highest degree values in the sample are exclusively popular American and French rap artists as shown in table 3.9.

k	artist	$B \times 10^6$	artist
243	T.I.	9.5	GEE FUTURISTIC
227	Timbaland	8.8	Eko du 94
215	DJ KHALED	7.6	X-plosive Beats Production
208	RIM'K	5.4	Beat Maker Naiche
204	BOOBA	5.3	Cool & Dre
192	Jay-Z	4.9	Aiko Rohd Infrarohd Music
183	oxmO.Puccino	4.5	DJ KHALED
179	The Game	4.0	Brisk Fingaz
178	J DILLA	3.7	Prince NegaaFellaga
170	DJ PREMIER	3.2	Akon

Table 3.9: Artists with highest degree and betweenness in the MySpace artist network sample

We see some more variation in the highest betweenness artists but the list is still dominated by "hip-hop" and "rap" artists. Many of the high betweenness artists are German (Gee Futuristic, Eko du 94, X-plosive Beats)

[15] http://www.myspace.com/timbaland

or Swiss (Beat Maker Naiche, Prince NegaaFellaga) artists. The listings in table 3.9 suggest that our network structure may be correlated with both the geographic locations of artists and artist genre. We can examine the assortativity coefficients to further test this assumption. A variety of assortativity coefficients for the MySpace artist network are presented in table 3.10.

r_k	r_{views}	$r_{friends}$	r_{plays}	$r_{country}$	r_{genre}	σ_r
-0.0326	0.0173	0.0132	0.0152	0.729	0.350	1.79×10^{-3}

Table 3.10: Assortativity coefficients for the MySpace top-friend artist network where r_k is assortativity with respect to degree, r_{views} is assortativity wrt number of profile views, $r_{friends}$ is assortativity with respect to total number of friends, r_{plays} is the assortativity wrt to plays, $r_{country}$ is assortativity wrt the country, and r_{genre} is the assortativity wrt to genre using a truncated genre list

In addition to collecting the top friends and genre labels for the artists in the MySpace artist network sample, we collect the artist's country, total number of profile views, total number of times an artist's tracks have been played and total number of friends (recall this is distinct from "top friends"). We can calculate assortativity coefficients for each of these properties to see how they relate to the network structure.

We see that the assortativity wrt degree r_k is actually very slightly negative indicating that artists have tendency to specify top friends who have *different* in-degree values. Intuitively, we might expect many obscure artists to include popular or influential artists in their top friend list - creating many links between low-degree artists and high-degree artists. Perhaps this phenomenon counter balances the gregarious node homophily commonly expected in social networks where popular (high-degree) people preferentially connect to other popular people [Newman, 2003a]. This results in a r_k value that is very close to that of a randomly mixed network where $r_k = 0$.

The assortativity coefficient for genre r_{genre} gives a positive value. To calculate the genre assortativity we truncate the list of 0-3 genres to include only the one genre listed first. Artists who specify no genre are excluded from the calculation. A more in depth analysis of the genre assortativity for this network sample and alternative approaches to calculating genre assortativity are presented in [Jacobson and Sandler, 2008]. The network sample clearly demonstrates assortative mixing with respect to genre indicating the network structure is relevant in the context of musical studies.

The assortativity coefficient for the artist's country $r_{country}$ is also positive,

with a rather high value of $r_{country} = 0.727$ indicating that artists have a strong tendancy to prefer to form connections with other artists in the same country. This shows the network structure is highly correlated with the country location of the artists.

We see slightly positive values for assortativity with respect to the total number of profile views, the total number of friends, and total plays. These properties can be considered a measure of popularity. In a pure social network, the degree of a node is the authoritative measure of popularity. However, because of our "top friends" sampling methodology and our criteria of only including artist nodes in our sample, we resort to these alternative measures of popularity assuming "top friend" artist popularity to be distinct from general popularity. We can use our popularity correlation coefficient to better measure the correlation of degree in the top-friend network to general popularity. We find values of $\rho_{views} = 0.22$, $\rho_{friends} = 0.27$, and $\rho_{plays} = 0.18$ showing degree in top-friend artist network is actually nominally correlated with our metrics for general popularity.

It should be noted that all three metrics for popularity in the myspace artist network are highly correlated. The Pearson correlation coefficient calculated between total friends and profile views has a value of $r = 0.80$ while the correlation between plays and profile views of $r = 0.71$. It is significant that these popularity measures have a much higher correlation between themselves than with the degree values in the network (measure by the popularity correlation coefficient ρ).

We argue that this supports our original assumption that the top friend relationship for a MySpace artist implies something slightly different from a social connection. If the top friend relation were simply equivalent to a social connection we would expect a stronger correlation between nodes degrees and popularity. The top friend relation for MySpace artists is perhaps closer to a music-related association such as collaboration or influence. Then we might infer that the overall degree of an artist in our sample relates to something like musical influence rather than raw popularity (which is better represented in terms of total friends, profile views, or total plays). We can examine the list of artists with the highest numbers of total friends and profile views in table 3.11 and the artists with the most plays in 3.12.

Many of the same names appear in all three lists which is expected given our three popularity metrics are highly correlated. The presence of the artist Tila Tequila[16] and other very popular artists suggests that our sample in fact

[16]Tila Tequila was the most popular artist on MySpace in terms of profile views in 2006 see http://www.slate.com/id/2139691/

views($\times 10^7$)	artist	total friends ($\times 10^6$)	artist
31.85	Akon	3.88	Tila Tequila
20.47	Lil Wayne	2.64	T.I.
20.06	Tila Tequila	1.90	Lil Wayne
11.80	T.I.	1.80	Rihanna
11.64	Chris Brown	1.74	Fall Out Boy
11.38	Colby O'Donis	1.63	Chris Brown
9.86	Colby O'Donis (2nd profile)	1.62	Beyonce
9.61	Rihanna	1.30	Avril Lavigne
8.45	Soulja Boy Tell 'Em	1.30	Soulja Boy Tell 'Em
8.02	Beyonce	1.26	50 Cent

Table 3.11: Artists with the most views and most total friends in the MySpace artist network

total plays $\times 10^7$	artist
3.82	Lil Wayne
3.49	T.I.
3.32	Akon
3.08	Jim Jones
2.99	My Chemical Romance
2.76	Nelly Furtado
2.57	Beyonce
2.52	Danity Kane
2.41	Pretty Ricky
2.35	Sean Paul

Table 3.12: Artists with the most plays in the MySpace artist network

does include many of the most popular artists on MySpace. Note the lists in table 3.11 and table 3.12 include only American pop and rap artists unlike the highest degree and betweenness lists in table 3.9 which also include many acts from continental Europe. This further supports the assertion that top-friend network structure is reflecting something other than raw popularity.

3.6.4 Audio-based Analysis

As mentioned in section 3.6.1 we also collect the audio files associated with each artist in our network. We use measures of audio-based similarity to build an alternative network of artists and compare this network to the original MySpace top-friends artist network.

A variety of methods have been developed for signal-based music analysis, characterizing a music signal by its timbre [Logan and Salomon, 2001], harmony [Bello, 2003], rhythm [Gouyon and Dixon, 2005], or structure [Levy et al., 2006]. We can apply some of these automatic methods to the audio associated with our artist nodes and ask: *Are audio-based music similarity measures correlated with the structure of the artist social network?* This question was first addressed in [Fields et al., 2008] where it was shown that the geodesic distance of two artists in the top-friend network is not correlated with the audio-based similarity scores.

One of the most widely used methods in audio-based music similarity is the application of Mel-frequency cepstral coefficients (MFCC) to the modeling of timbre [Logan, 2000]. In combination with various statistical techniques, MFCCs have been successfully applied to music similarity and genre classification tasks [Tzanetakis and Cook, 2002a; Pampalk, 2006; Logan and Salomon, 2001; Berenzweig et al., 2004; Jacobson, 2006]. MFCCs are a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency. In this sense they represent the "frequency of frequencies". MFCCs are actually intended to be pitch invariant and most closely represent the timbre of an audio signal.

We apply the approach outlined by [Tzanetakis, 2009] to determine a set of audio-based similarity measures for the MySpace artist network. In addition to MFCCs this method incorporates spectral centroid, spectral rolloff, and spectral flux features. The spectral centroid is defined as the center of gravity for the spectral magnitudes of a frame of the short-time Fourier transform. The spectral rolloff is defined as the frequency below which 85% of the magnitude spectrum is concentrated. Spectral flux is defined as the squared difference between normalized magnitudes of successive spectral distributions. Additional details about these features are available in [Tzanetakis and Cook, 2002b]. These features are concatenated and a running mean and standard deviation is calculated of the previous $M = 40$ frames resulting in a sequence of 32-dimensional feature vectors. Finally the mean and standard deviation is calculated across these feature vectors to create a single 64-dimensional feature vector representing a given audio file. A simple Euclidean distance is used to determine the similarity between any pair of audio files. This approach has been shown to perform well in the MIREX[17] audio similarity task consistently scoring among the top entries. We make use of the open-source MARSYAS[18] audio similarity implementation.

[17] see http://www.music-ir.org/mirex/ for a history of competition results

[18] see http://marsyas.info/

Each artist might have several audio files associated with their MySpace profile. If this is the case, we select the audio file that has received the most plays as representative of the artist. Note that for 238 artists the audio was either missing or corrupted. These artists were left out of the audio-based analysis. Given a list of representative songs for each artist, we calculate the pair-wise artist similarity distance using the approach from [Tzanetakis, 2009]. A new graph is constructed by first removing every edge in our original top-friend MySpace network sample, setting a threshold value, and creating an edge between each pair of artists whose inter-artist distance is below the threshold. The threshold value was selected to obtain a network with approximately the same number of edges as were present in the original MySpace artist network. For our network sample a threshold of $t = 0.395$ on the inter-artist audio distances resulted in a network with $m = 127,824$ edges - slightly more than the $m = 120,487$ edges found in the top-friends network sample.

However, the audio-based artist network is drastically different in almost every other respect. Some of the network statistics for the audio-based similarity MySpace artist network are provided in table 3.13.

n	m	n_{S_0}	$\|S\|$	$\langle n_S \rangle$
15,240	127,802	6,552 (43.0%)	8,539	1.81

Table 3.13: Table of network statistics for the audio-based similarity MySpace artist network where n is the number of artist nodes, m is the number of edges, n_{S_0} is the number of nodes in the largest connected component, $|S|$ is the total number of components and $\langle n_S \rangle$ is the average number of nodes in each connected component in the network.

Because the audio-based similarity measure is symmetric, our new network is undirected. The audio-based similarity network is highly fragmented - of the $8,539$ components 96.4% of them consist of single nodes. These types of outliers are common in audio-based similarity measures. Some audio signals are found to have a very low similarity to other signals while some signals actually act as similarity hubs [Aucouturier and Pachet, 2008]. However, we see a degree distribution for the audio-based similarity network that better approximates an exponential distribution as indicated by the log-normal plot in figure 3.5.

In table 3.14 we see some statistics related to the largest connected component of the audio-based similarity network.

The average shortest path and network diameter are both larger than

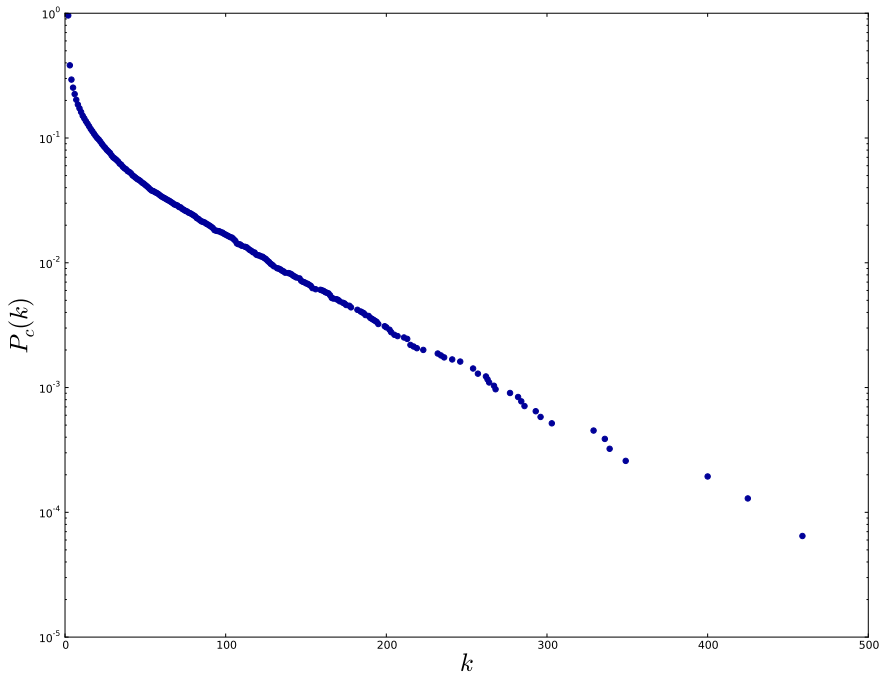


Figure 3.5: Cumulative degree distributions for the MySpace audio-based similarity artist network

n	m	$\langle d \rangle$	d_{max}	T
6,552	112,462	4.65 (2.82)	23 (4)	0.27 (5.0×10^{-3})

Table 3.14: Network statistics for the largest connected component of the MySpace audio-based similarity artist network where n is the number of nodes, m is the number of edges, $\langle d \rangle$ is the average geodesic distance, d_{max} is the diameter and T is the clustering coefficient.

that of an equivalent random network - a stark divergence from the small-world nature of the top-friends network. The audio-based network does have a transitivity value that is significantly higher than that of an equivalent random network.

We see some of the most interesting statistics in the assortativity measures for the audio-based network shown in table 3.15.

All of the assortativity measures are positive. Interestingly we see sig-

r_k	r_{plays}	r_{views}	$r_{friends}$	$r_{country}$	r_{genre}	σ_r
0.214	0.216	0.248	0.192	0.314	0.252	2.68×10^{-3}

Table 3.15: Assortativity coefficients for the MySpace audio-based artist network where r_k is assortativity with respect to degree, r_{plays} is assortativity wrt number of song plays, r_{views} is assortativity wrt number of profile views, $r_{friends}$ is assortativity with respect to total number of friends, $r_{country}$ is assortativity wrt the country, and r_{genre} is the assortativity wrt to genre using a truncated genre list

nificantly higher assortativity with respect to our popularity metrics than observed in the MySpace top-friends network. The values of r_{views} and $r_{friends}$ are both an order of magnitude larger than their respective values in the top-friends network. At first this seems counter-intuitive: we would expect the audio-based similarity network to be *less* correlated with popularity. However, if we examine our popularity correlation coefficient for the audio-based network, this is exactly what we find: $\rho_{plays} = -7.0 \times 10^{-3}$, $\rho_{views} = -5.1 \times 10^{-3}$ and $\rho_{friends} = 6.3 \times 10^{-3}$ show that degree in the audio-based network is completely orthogonal to our popularity metrics. Yet, we have some clear assortativity with respect popularity in the audio-based network. What we see then, is that popular artists have a tendency to *sound similar* (at least by our audio-based metric) to other popular artists. Popular artists perhaps tend to use the same timbres and production techniques employed by other popular artists causing popular artists to be preferentially similar to other popular artists. Likewise, unpopular artists might use lower-quality production techniques and lack a mastering stage in their production process, causing unpopular artists to sound like other unpopular artists. These findings give some credence to the gripe music purists often make: "All popular music sounds the same!"

This correlation between the structure of an audio-based artist similarity network and artist popularity runs contrary to the results reported by Celma [2008] where an artist network constructed using audio-based similarity techniques was shown to have almost no correlation with popularity metrics. For Celma's network $r_{plays} = 0.080$ and $r_{genre} = 0.089$. Additional statistics reported by Celma are re-printed in table 3.26. Celma's network was considerably larger with $n = 59,583$ and $m = 1,79,743$ and a different audio-based analysis was applied [Cano et al., 2005b] against a much larger corpus of audio files (1.3 million audio files compared to just over 15 thousand). Furthermore, Celma uses play count data from last.fm which is distinct from play counts collected on MySpace (used in our analysis) and Celma assigns

genre labels based on a last.fm tag aggregation following [Sordo et al., 2008] while we simply use MySpace genre labels. It is not clear which of these discrepancies is responsible for the disparity in network structures.

In the MySpace audio-based network we see significant assortativity with respect to country and genre although both values are lower than the respective values in the top-friends network.

Finally we can simply examine the list of the highest degree artists in the audio-based artist network shown in table 3.16 and see that these names appear in neither the top-friends, highest views, highest plays or highest total friends lists. We can be more quantitative by calculating the Pearson

k	artist name
576	Kevin Reveyrand
562	Deonna Martin
560	Grizzly
487	DJ Sense
475	Seesha
457	La Dinastia Nueva Cancion
454	noar
448	Kidrass
443	Mioritza
433	9th Wonder

Table 3.16: Highest degree artists in the MySpace audio-based similarity network

correleation between the degree values in the top-friend network and the degree values in the audio-based similarity network. Doing so results in a value very near zero $r = 4.1 \times 10^{-3}$ suggesting the degree values in the two networks are orthogonal. In short, other than sharing the same set of nodes, the audio-based artist network and the top friends artist network have very distinct structures.

However, we must qualify our findings with a brief discussion of lossy audio encodings. The audio files collected from MySpace are encoded in a uniform format - 96kbps MP3. However, MySpace users may upload files in a variety of formats and therefore the audio presumably under goes some transcoding process. It has been shown that MP3 encoding at a rate of 96kbps hinders the accuracy of an audio-based classification task when compared to the same task on lossless encoded audio [Jacobson et al., 2008a]. In Jacobson et al. [2008a] the degradation in classification performance reported

for a 96kbps encodings was small but statistically significant. Furthermore, a set of heterogeneous encodings caused even greater degradation of classification performance. However, we are applying a more advanced audio-based analysis here and we are performing a similarity task rather than a classification task. It seems unlikely that the use of moderately low-bit-rate audio encodings could somehow artificially induce a network structure with the assortativity characteristics we have observed.

3.6.5 MySpace Summary

We have presented two distinct views of the MySpace artist network - one where the edges represent the directed "top-friend" relationship as specified on the MySpace website and a second where the edges represent an inter-artist audio-based similarity. The two networks are almost entirely orthogonal. The MySpace top-friend network is small-world and exhibits a multi-scaling degree distribution that closely follows a power-law for moderate degrees. Notably, the top-friend network exhibits slightly dissassortative mixing with respect to degree - an interesting divergence from the structures expected in social networks. The MySpace audio-based similarity network is highly fragmented, not small-world, and exhibits an exponential degree distribution. Furthermore, the audio-based similarity network does exhibit a nominal level of assortative mixing with respect to degree.

3.7 Soundcloud Artist Network

Soundcloud[19] provides a platform for sending, receiving, and distributing audio files. Where MySpace is an online social network that became a vehicle for music artist promotion Soundcloud was purpose-built for musicians, record labels, and music producers. Users can upload audio content to share privately or publicly. A user can also "follow" another user to receive notifications and updates about that user's activity. This creates a directed social network of music artists similar to the MySpace top-friend artist network.

The Soundcloud artist network was crawled during the week of July 13th 2009 by collecting all the data available through the site's newly released API[20]. This includes each user's list of followers, country, city, and user

[19] see http://soundcloud.com/

[20] see http://soundcloud.com/developers for details on the API, Ben Fields was responsible for the original crawl

name. Later in February of 2010 additional information about each artist was collected via web-scraping including the number of track plays and number of track downloads. We summarize the properties of the Soundcloud artist network in table 3.17.

n	m	n_{S_0}	$\|S\|$	$\langle n_S \rangle$
285,002	2,477,999	107,802 (37.8%)	174,882	1.24

Table 3.17: Network statistics for the Soundcloud artist network where n is the number of artist nodes, m is the number of release edges, n_{S_0} is the number of nodes in the largest connected component, $|S|$ is the total number of components and $\langle n_S \rangle$ is the average number of nodes in each connected component in the network.

We see that the Soundcloud artist network is highly fragmented. There are many solitary artist nodes and many very small components with an average component size of $\langle n_S \rangle = 1.24$. This suggests many users ignore the social networking aspect of the site and only use the audio sharing functionality. In this sense the Soundcloud network is quite different from the MySpace network where the specification of "friendship" is the primary function. However, in almost every other respect the network structure of the Soundcloud network is very similar to that of the MySpace top-friend network.

The cumulative degree distribution is plotted in figure 3.6. In the log-log plot we can clearly see a very close approximation of a power-law as the distribution follows a straight line for both the in-degree and the out-degree. Again this demonstrates the existence of hub artists who have many more followers than the average. Note the brick-wall cutoff of the out-degree distribution resulting from the limitation of $k_{out} \leq 2000$ imposed by a maximum number of artists a given user can follow.

In the Soundcloud network each artist node has a number of plays indicating the total number of times the artist's tracks have been played, a total number of track downloads indicating how many times an artist's tracks have been downloaded, and a label indicating the artist's country. We can use these attributes to calculate assortativity coefficients for the Soundcloud network. The results of the assortativity calculations are shown in table 3.18.

Again, we see clear assortative mixing with respect to country - artists more often tend to follow other artists from the same country. This same mixing pattern was seen in the MySpace artist network but to a greater extent (for the MySpace artist network $r_{country} = 0.729$). Again, similar to the

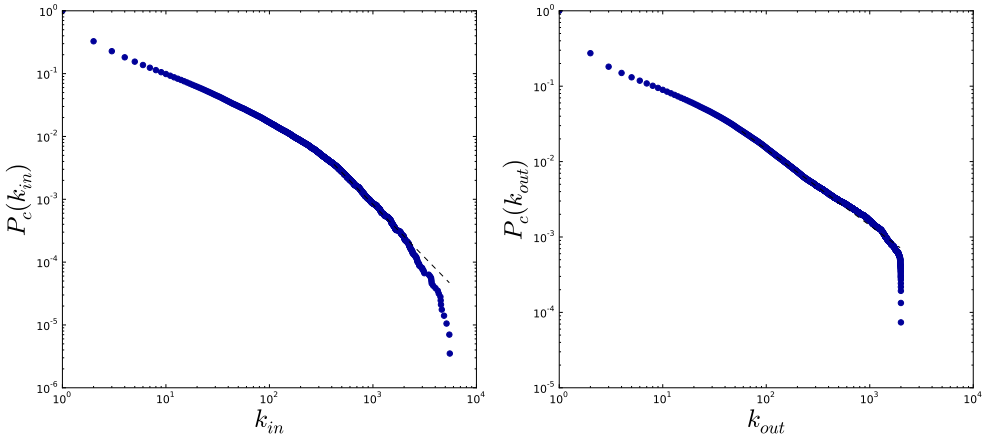


Figure 3.6: The cumulative degree distributions for the Soundcloud artist network with (a) the in-degree cummulative distribution plotted on a log-log scale showing a power-law fit with $\alpha = 2.69$ and (b) the out-degree cummulative distribution plotted on a log-log scale showing a power-law fist with exponent$\alpha = 2.01$

r_k	r_{plays}	$r_{downloads}$	$r_{country}$	σ_r
-0.137	3.9×10^{-3}	0.0113	0.201	3.9×10^{-3}

Table 3.18: Assortativity coefficients for the Soundcloud artist network where r_k is assortativity with respect to degree, r_{plays} is assortativity with respect to number of plays, $r_{downloads}$ is assortativity with respect to total number of downloads, $r_{country}$ is assortativity with respect to country, and σ_r is the error in the assortativity coefficient calculation.

MySpace network, in the Soundcloud artist network we see slightly *dissassortative* mixing with respect to degree. This divergence from the assortative degree mixing expected in pure social networks [Newman, 2003a] suggests that the "following" relationship in the Soundcloud network is perhaps more indicative of influence rather than friendship. As mentioned in the context of the MySpace top-friend network, the dissassortative degree mixing in the Soundcloud network is likely the result of lower-degree artists having a tendency to follow high-degree artists who are well-known and influential.

For the Soundcloud network, we have two scalar attributes that serve as indicators of artist popularity - the number of times an artist's tracks have been downloaded, and the number of times an artist's tracks have been played

streaming. Note that Soundcloud artists have the option to *not* make their audio material available for download and many artists exercise this option preferring to only offer streaming audio. In this sense the number of plays is the more appropriate measure of popularity because it applies to nearly all artists in the network. We see that the value for assortativity with respect to plays $r_{plays} = 3.94 \times 10^{-3}$ is slightly positive but almost equivalent to the error $\sigma = 3.90 \times 10^{-3}$. The value for assortativity with respect to downloads is an order of magnitude larger, but still relatively close to zero. These values suggest that the Soundcloud network is randomly mixed with respect to popularity. Furthermore, the popularity correlation coefficient for both popularity metrics is close to zero with $\rho_{downloads} = 0.026$ and $\rho_{plays} = 0.055$ showing that the node in-degree (number of followers) is orthogonal to these popularity metrics. As in the MySpace artist network, the combination of popularity-based assortativity near zero and negative values for r_k suggests that the Soundcloud artist network structure is more closely tied to some notion of musical influence rather than general popularity.

3.8 Echo Nest Artist Network

Co-founded by music informatics researchers Brian Whitman and Tristan Jehan, Echo Nest[21] is a company providing a variety of services based on a machine learning platform for music. As stated on the website Echo Nest "combines large-scale data mining, natural language processing, acoustic analysis and machine learning to automatically listen to music, read about music, and learn music trends (on the web)." Echo Nest provides a dizzying array of music informatics services through a web API[22] including artist information, high-level audio-based song analysis, and listener demographics. We focus on the artist similarity data provided by Echo Nest which allows us to construct yet another artist network.

A sample of the Echo Nest artist similarity dataset[23] was made available by the Echo Nest as part of Paul Lamere and Justin Donaldson's tutorial on visual interfaces for music collection navigation at the ISMIR conference in October 2009. This dataset contains approximately 70,000 artists (nodes) and 270,000 similar artist relations (edges). We provide a summary of the network statistics for the Echo Nest artist network in table 3.19.

We see the network contains some fragmented components. Upon closer

[21] see http://echonest.com/

[22] see http://developer.echonest.com/

[23] available at http://sites.google.com/site/musicviz2/datasets

n	m	n_{S_0}	$\|S\|$	$\langle n_S \rangle$
68,527	268,734	52,643 (76.8%)	15,870	1.35

Table 3.19: Network statistics for the Echo Nest artist recommendation network including the number of nodes n, the number of edges m, the number of nodes in the largest connected component n_{S_0}, the total number of components $|S|$ and the average number of nodes in each connected component in the network $\langle n_S \rangle$.

examination we see that all but two of these components contain only one artist node. The second largest component contains only 15 artist nodes. This is likely a result of the provided data being truncated. The data is provided as a list of artists and associated metadata followed by a list of edges. It is likely not all the edges intended were included in the final output. We will therefore focus on the largest connected component assuming the fragmentation is an artifact of some truncation of a much larger data set. Some network statistics for the largest connected component are provided in table 3.20. We see many of the features we have come to expect - small

n	m	$\langle d \rangle$	d_{max}	T
52,643	268,719	7.22 (6.85)	19 (14)	0.157 (1.9×10^{-4})

Table 3.20: Network statistics for the largest connected component of the Echo Nest artist recommendation network n is the number of nodes, m is the number of edges, $\langle d \rangle$ is the average geodesic distance, d_{max} is the diameter, and T is the clustering coefficient.

worldness in terms of average geodesic distance and diameter and a high clustering coefficient.

By design, every artist node (with the exception of the singleton nodes) has an out-degree of $k_{out} = 15$. The cumulative distribution of the in-degree is plotted in figure 3.7. Instead of the more common power-law distribution, we see an exponential distribution indicated by the approximately straight line distribution appearing when plotted on the log-normal scale. If we assume that Echo Nest indexes less than 21 million artists, we might infer that we have met the 0.25% threshold specified by Kwak et al. [2006] and believe we have accurately estimated the degree distribution for the entire Echo Nest artist network. However, the sampling method employed to collect this data was not made explicit and the fragmented nature of the network tells us that a snowball sampling was not the method used (or if snowball sampling was

used the results were somehow truncated). Therefore we cannot make any conclusions about the degree distributions of the entire network because even if we assume we have an appropriate fraction of nodes, we cannot know if we have all the corresponding edges in our sample. And as stated before, the high number of singleton components suggest we are actually missing some portion of relevant edges for the Echo Nest sample.

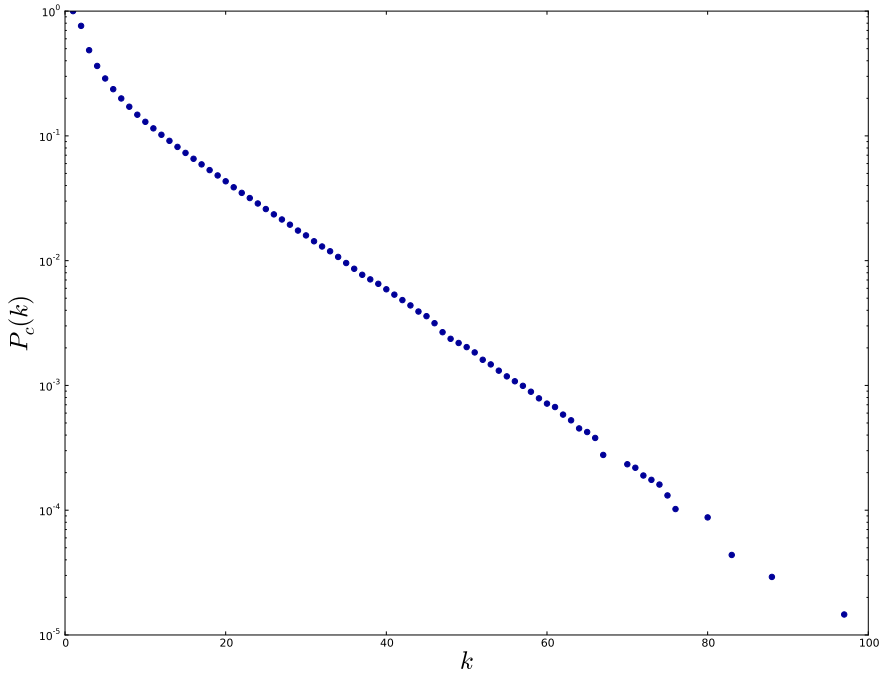


Figure 3.7: The cumulative in-degree distributions for the Echo Nest artist network plotted on a log-normal scale

But consider that an exponential distribution - lacking in hubs with extraordinarily high degree values - might be a beneficial characteristic for a recommendation network. A recommendation network with a power-law distribution would have hub artists that appear in recommendations much more often than other artists. For the end user, this could result in recommendations that are cyclical or uninteresting with the same hub artists appearing over and over again - a phenomenon explained in depth by Celma [2008]. A network with an exponential distribution would, to some extent, avoid such problems. Whether this is actually a design principal of the Echo Nest artist recommendation network is not disclosed.

In addition to the artist similarity relations, the dataset also includes *hotttness* and *familiarity* attributes for each artist. These are proprietary scalar metrics created by Echo Nest. The hotttness metric is a "numerical description of how hottt (sic) an artist currently is" - presumably based on the mining of time-varying data from the web. The familiarity metric is a "numerical estimation of how familiar an artist currently is to the world" - again presumably based on data from the web and for our purposes we will assume the familiarity scalar roughly corresponds to an artist popularity metric. We can use these attributes for assortativity calculations.

We also collect additional data about each artist from Last.fm (we discussed the Last.fm artist recommendation network in section 3.3.2). For each artist we collected the number of *plays*, the number of unique *listeners*, and the most commonly applied *tag* from the Last.fm API[24]. The plays and listeners count give us some indicator of an artist's popularity. The artist's tags give us some idea about the artist's genre. Although more sophisticated methods of mapping tags to genres have been applied [Sordo et al., 2008] we simply select the most commonly applied tag for the artist as representative for that artist. We apply all these artist node attributes to assortativity calculations and the results are shown in table 3.21. We see that this network

r_k	r_{plays}	$r_{listeners}$	$r_{familiarity}$	$r_{hotness}$	r_{tag}	σ_r
0.232	0.203	0.291	0.177	0.171	0.444	2.1×10^{-3}

Table 3.21: Assortativity coefficients for the Echo Nest artist network where r_k is assortativity with respect to degree, r_{plays} is assortativity with respect to number of plays, $r_{downloads}$ is assortativity with respect to total number of downloads, $r_{country}$ is assortativity with respect to country, and σ_r is the error in the assortativity coefficient calculation.

does exhibit assortative mixing with respect to degree contrary to most of the other artist networks we have discussed. We see the strongest assortative mixing is related to the top tag $r_{tag} = 0.444$. Despite having just over 3,000 distinct top tags in our Echo Nest sample, we still see strong assortative mixing. We also see assortative mixing for our popularity indicators r_{plays} and $r_{listeners}$. The weakest assortative mixing is seen for the familiarity and hotttness attributes although these values are still significantly positive at $> 80 * \sigma$.

Together these assortativity measures suggest that the structure of the Echo Nest artist similarity network is at least nominally correlated with

[24] see http://last.fm/api/

notions of artist popularity - popular artists tend to be connected to other popular artists. We can also examine the popularity correlation coefficients listed in table 3.22.

ρ_{plays}	$\rho_{listeners}$	$\rho_{familiarity}$	$\rho_{hotness}$
0.297	0.480	0.717	0.453

Table 3.22: Popularity correlation coefficients for the Echo Nest artist similarity network.

We can see that the degree values for the Echo Nest artist similarity network are most strongly correlated with proprietary familiarity metric. Artists that are considered familiar are recommended most often, perhaps by design. A positive correlation is seen for all of the available popularity metrics.

The Echo Nest artist similarity network sample is a small-world network that follows an exponential degree distribution (possibly by design, possibly because of truncation). Its structure is moderately to strongly correlated with the available popularity metrics.

We can also apply Celma's long-tail assortativity analysis described in section 3.3.2 to the Echo Nest network. We divide artist nodes into either the "head", "mid", or "tail" sections based on a given popularity metric (here we follow Celma and use Last.fm play counts) and apply an appropriate label. We can now calculate assortativity with respect to these long tail labels. We find that the Echo Nest artist similarity network exhibits rather strong assortativity with respect to long tail popularity with $r_{lt} = 0.453$. While the Echo Nest artist network sample avoids a power-law degree distribution it seems to suffer from some of same popularity bias that Celma found in the Last.fm artist recommendation network.

3.9 MusicBrainz artist network

MusicBrainz[25] is a user-maintained open community that collects, edits, and publishes music metadata on the web. The MusicBrainz project was started by Robert Kaye in 2000 as a response to a set of restrictive licensing terms imposed by CDDB[26]. The data collected by CDDB was largely crowd-sourced

[25] see http://musicbrainz.org/

[26] CDDB is a data service for identifying compact discs, see http://www.gracenote.com/

data. MusicBrainz was created to provide an alternative where contributors could be sure that open licensing terms would be applied to their contributions. MusicBrainz evolved into a very rich data service that provides a wide array of music-related metadata currated by an active community of contributors.

3.9.1 Identifiers

One of MusicBrainz most appealing features is its use of unique identifiers for music artists, albums, recordings, musical works, and record labels. MusicBrainz generates identifiers using the universally-unique identifier (uuid) approach [Leach et al., 2005]. While the 36 character codes that make up a uuid are not particularly easy for humans to parse, they allow applications to deal with music metadata unambiguously. For example, there exists several music artists named James Brown - James Brown "The Godfather of Soul", James Brown the Canadian drummer for Veil Marker, James "Razors" Brown the indie-rock recording engineer, James Brown the British Oboe player, and several others. An application that attempts to use the name "James Brown" as an identifier for an artist will potentially run into problems. However, an application that uses MusicBrainz identifiers will know that James Brown "The Godfather of Soul" identified by `20ff3303-4fe2-4a47-a1b6-291e26aa3438` is distinct from James "Razors" Brown identified by `206028fc-e674-40b6-b1b5-c60bd9e12455`. In section 4.6.1 we will discuss how these identifiers are useful for creating URIs for music-related entities.

3.9.2 Advanced Relationships

MusicBrainz also includes a set of Advanced Relationships that enable the description of inter-connections between music-related entities. For example, it is possible to specify that two artists are married to each other or that they share some other familial relationship. It is also possible to specify that a recording is a performance of a particular work or that one recording samples another recording. As of this writing, there are just over 300 Advanced Relationship types in the MusicBrainz database.

3.9.3 The Six Degrees Application

The *Six Degrees of Black Sabbath*[27] web application created by Paul Lamere of Echo Nest leverages the Advanced Relationships in MusicBrainz to find paths between music artists. Users enter a pair of artist names and the application finds a path between the two artists following a wide range of artist relations.

In addition to including artist-to-artist relations for MusicBrainz, the application collapses artist-to-album, artist-to-track, and track-to-track relationships into artist-to-artist relationships. The result is a directed artist network with 95 distinct edge types - a *multiplex* network. We present a brief analysis of this network and save a more complete analysis of MusicBrainz Advanced Relationships for future work. Some of the network statistics for the Six Degrees artist network are summarized in table 3.23:

n	m	n_{S_0}	$\|S\|$	$\langle n_S \rangle$
219,246	1,092,085	186,509 (85.1%)	10,128	21.65

Table 3.23: Network statistics for the MusicBrainz Six Degrees artist network where n is the number of artist nodes, m is the number of release edges, n_{S_0} is the number of nodes in the largest connected component, $|S|$ is the total number of components and $\langle n_S \rangle$ is the average number of nodes in each connected component in the network.

The Six Degrees artist network has one large connected component with many smaller connected components. Excluding the giant component, all components have less than 75 artist nodes. Some statistics for the largest connected component are presented in table 3.24.

n	m	$\langle d \rangle$	d_{max}	T
186,509	356,179	7.29 (9.17)	29 (19)	6.6×10^{-3} (1.98×10^{-5})

Table 3.24: Network statistics for the largest connected component of the MusicBrainz Six Degrees artist network n is the number of nodes, m is the number of edges, $\langle d \rangle$ is the average geodesic distance, d_{max} is the diameter, and T is the clustering coefficient.

The giant connected component has a rather large diameter and a very

[27] see http://labs.echonest.com/SixDegrees

low clustering coefficient - diverging somewhat from the small-world network model. However, it does exhibit a relatively small average geodesic distance.

It can be seen that both the in degree and out degree distributions for the Six Degrees artist network approximate a power law by examining the distributions in figure 3.8.

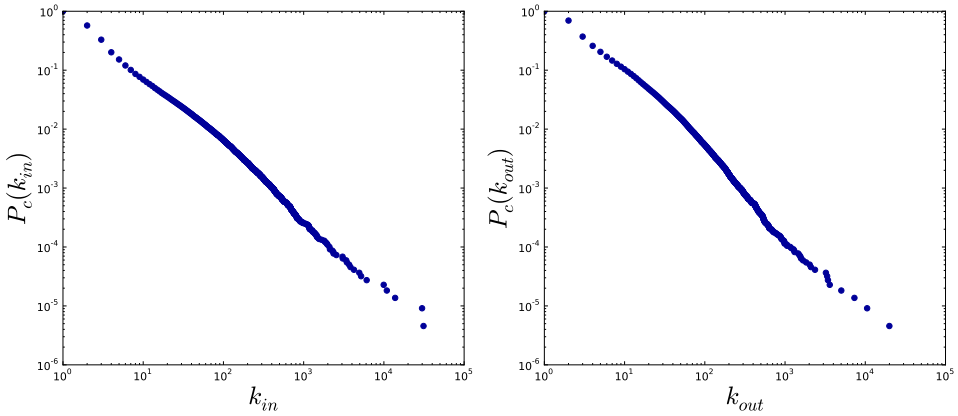


Figure 3.8: The cumulative degree distributions for MusicBrainz artist network.

The artists in the MusicBrainz Six Degrees network are also matched against Last.fm to obtain play count, number of listeners, and top tag data. However, less than half of the artists in the Six Degrees network were successfully matched to Last.fm (for the Echo Nest artist network that also includes data from Last.fm matching, a success rate of 96% was achieved). This is presumably because Last.fm does not seem to utilize MusicBrainz identifiers for more obscure artists. The Six Degrees network is an order of magnitude larger than the Echo Nest artist recommendation network sample and presumably contains many more obscure artists.

We still calculate assortativity values related to the Last.fm alignment by ignoring unmatched nodes. These values are reported in table 3.25 but should be treated with some trepidation.

We see what is essentially random mixing with respect to degree (the value of r_k is of course unaffected by deficiencies in the Last.fm artist matching). We also see that the portion of the network that is matched exhibits nearly random mixing with respect to play counts and number of listeners. There is some minor assortative mixing with respect to top tag. The network exhibits some small positive correlation between play count and degree with $\rho_{plays} = 0.182$.

r_k	r_{plays}	$r_{listeners}$	r_{tag}
-0.0199	0.0114	0.0205	0.187

Table 3.25: Assortativity coefficients for the MusicBrainz Six Degrees artist network where r_k is assortativity with respect to degree, r_{plays} is assortativity with respect to number of plays, r_{tag} is assortativity with respect to most frequently applied tag, and σ_r is the error in the assortativity coefficient calculation. It should be noted that values for r_{plays}, $r_{listners}$, and r_{tag} are based on calculations using only 46% of the total nodes due to deficiencies in matching artists to Last.fm

3.10 Summary of Artist Network Analysis

We have reviewed the structure of several artist networks found on the web. We examined two expert curated resources - the Classical Music Navigator (original analysis) and the All Music Guide (a survey of previous work). We see that hubs exist in the composer influence network as well as in the AMG artist collaboration network. However, the AMG artist similarity network follows an exponential degree distribution lacking hubs. In the AMG network we see a variety of relationships that can exist between artists - collaboration, similarity, influence, and membership. This multiplex of edge types mirrors the reality of inter-artist associations - a wide variety of inter-artist connections are possible. The MusicBrainz database contains 13 distinct types of artist-to-artist relationships and over 305 types of relationships when artist, recording, release, and label nodes are included - again reflecting the diversity of possible connections found between musical entities. In this chapter, we have shown that for artist networks, the source of the inter-artist connections and the type of the inter-artist connections has a measurable effect on the network structures that arise.

The Discogs artist-release network was highly fragmented - consisting of one large connected component and many smaller components. It also followed a scale-free degree distribution. The other crowd-sourced artist networks - the MySpace artist network and the Soundcloud artist network - also approximated power-law distributions. In the artist recommendation networks from The Echo Nest and Last.fm we see an exponential decay and a power-law decay respectively for degree distributions - an interesting result that could have implications for the utility of these networks. However, both network structures seem to have significant correlations with artist popularity measures and have relatively few inter-artist connections crossing from

the popularity head to the long tail.

We have seen values very near zero or slightly negative values for r_k in most of the networks we have examined including the classical composer influence network, the Discogs artist-release network, the MySpace artist network, and the Soundcloud artist network. This means there is no clear tendancy for high-degree nodes to prefferentially connect to other high-degree nodes. Now consider that the MySpace network and (perhaps to a lesser extent) the Soundcloud network are both *social networks* - the edges imply "friendship" in some broad and loosely defined sense and these edges are created by the artists themselves. It has been reported that social networks usually have *positive* degree correlations and therefore *positive* values for r_k [Newman, 2003a]. These artist networks are exhibiting a structure that is distinct from most social networks. This is perhaps because artists maintain social relationships differently than normal people. We are using the term "social relationship" very broadly here. The term "following" used by the Soundcloud website is perhaps more appropriate than the term "top friend" used in MySpace. The lack of assortative mixing with respect to degree suggests that music artists use these connections to "cite" influences as well as for specifying collaborators or friendships. The result is there exists many connections between low-degree artists and high-degree artists which balances the homophily tendancy of the social component. This means in addition to a social element, these networks have an information component.

Whenever data was available, we have seen assortative mixing with respect to country. Perhaps not surprisingly, artists' choice of other artists to form relationships with seems to be highly influenced by nationality. This is probably because shared culture and language and geographic proximity tends to correlate with nationality.

Similarly whenever genre or genre-like data (tags) were available we see assortative mixing. This confirms that these network structures have some grounding in traditional music-related concepts.

Using audio-based analysis on audio from the MySpace artists an artist similarity network was also constructed. The structure of this network was not only distinct from that of the MySpace top-friend but also different from other audio-based similarity networks described in the literature.

For each network examined, music artists are the nodes. Different sources of inter-artist connections are used to create the network edges. Depending on the type and the provenance of the inter-artist connections used to create the network, we see a variety of distinct network structures emerge. In this sense we see the folly of treating an inter-artist connection as a one-

dimensional property. In chapter 5 we develop a framework that models the diversity and complexity of inter-entity connections in a uniform way but first we will discuss semantic web technologies and how they have been applied to music informatics in chapter 4.

net name	n	m	n_{S_0}	$\|S\|$	$\langle d \rangle$	d_{max}	$T(T_{random})$	α	r_k	r_γ	r_{lt}	r_{plays}	ρ_{plays}
last.fm	122,801	1,735,179	-	-	5.64 (4.42)	10	0.230 (1×10^{-4})	2.31	0.920	0.343	0.397	0.503	-
celma cb	59,583	1,179,743	-	-	4.48 (4.30)	7	0.025 (2.0×10^{-4})	1.61	0.140	0.089	-0.032	0.080	-
amg sim	74,494	407,483	-	-	5.92 (6.60)	9	0.027 (7.0×10^{-5})	exp	0.17	0.411	-0.002	0.259	-
cmn	426	1,780	426(100%)	1	3.39(3.06)	10 (6)	0.144 (0.019)	exp	-0.161	0.396	0.386	-0.0382	0.566
discogs	125,498	476,931	62,046(49.4%)	25,647	9.04(8.34)	31(17)	$0.110(4.0 \times 10^{-5})$	2.83	0.066	-	-	-	-
myspace	15,478	120,487	15,478(100%)	1	6.43(4.93)	16 (10)	0.219 (7.1×10^{-4})	3.39	-0.0326	0.350	0.480	0.0152	0.182
mysp audio	15,240	127,802	6,552(43.0%)	8,539	4.65(2.82)	23(4)	0.27 (5.0×10^{-3})	exp	0.214	0.242	0.529	0.216	7.0×10^{-3}
soundcloud	285,002	2,477,999	107,802(37.8%)	174,882	4.12	15(7)	0.120 (6.34×10^{-5}	2.69	-0.137	-	0.301	3.9×10^{-3}	0.055
echonest	68,527	268,734	52,643(76.8%)	15,870	7.22(6.85)	19(14)	0.157(1.9×10^{-4})	exp	0.232	0.444	0.493	0.203	0.297
musibrainz	219,246	1,092,085	186,509 (85.1%)	10,128	7.29 (9.17)	29 (19)	6.6×10^{-3} (4.82×10^{-5})	1.39	−0.0199	0.187	0.397	0.0114	0.182

Table 3.26: A summary of statistics for all networks discussed where last.fm refers to the Last.fm artist similarity network as reported by Celma [2008], celma cb refers to an audio-based similarity network as reported by Celma [2008], amg sim refers to the All Music Guide artist similarity network as reported by Celma [2008], cmn refers to the Classical Music Navigator composer influence network, discogs refers to the Discogs artist-release network, myspace refers to the MySpace top-friend network, mysp audio refers to the MySpace audio-based analysis network, soundcloud refers to the Soundcloud followers network, echonest refers to the Echonest and musicbrainz refers to the MusicBrainz Six Degrees artist network. Here n is the number of artist nodes, m is the number of release edges, n_{S_0} is the number of nodes in the largest connected component, $|S|$ is the total number of components, $\langle d \rangle$ is the average geodesic distance, d_{max} is the diameter, T is the clustering coefficient, α is the power-law coefficient, r_k is the assortativity coefficient with respect to degree, r_γ is assortativity with respect to genre or top tag, r_{lt} is assortativity with respect to long tail position, r_{plays} is assortativity with respect to total play counts, and ρ_{plays} is the popularity correlation coefficient for the network based on play counts. Values in parenthesis indicate values for an equivalent random network except for n_{S_0} where the value in parenthesis is the percentage of nodes from the entire network in the largest connected component.

Chapter 4

Music and Semantic Web Technology

> *Consciousness is connected with one unity. A machine is composed of parts.*
>
> –Kurt Gödel, as recorded in *A Logical Journey*, 1996

As we saw in chapter 3 there is a lot of structure inherent in the music-related data that can be found on the web. However, for the most part, this structure is somewhat obfuscated - especially from the perspective of the machine. Much of this structured data is formatted for human consumption - as pictures and text - and not provided in a format that is easy for a computer program to parse.

To illustrate this point, let us suppose, whether out of morbid musical curiosity, some socio-musicological undertaking or perhaps simply to settle some bet, we want to answer the following query:

> *Show me a list of deceased jazz piano players and their respective causes of death.*

Surely the data to satisfactorily answer this query exists somewhere on the web. In fact the information needed to answer this question could likely be found on one website - Wikipedia. However, to answer the query we would have to visit thousands of webpages on that site, scanning the text of each page to find information about cause-of-death, instruments-played, and musical genre. Of course, this is the sort of task we would rather have a

computer program perform. Unfortunately, with traditional web technologies the computer can not answer this query, at least not without some considerable programming effort. This is simply because the data on Wikipedia is not easy for a computer program to parse. The data is more-or-less free text intended for human consumption. Even the database software backing Wikipedia would be of little help answering this query - the data modeling approach used in the software powering Wikipedia focuses on managing users and page edits not on the actual meaning of the content. Furthermore, the relational database powering Wikipedia is not exposed on the web - we can not make some arbitrary Structured Query Language (SQL) query against the database and to expect that we should be allowed to do such a thing would seem ludicrous from a security perspective.

It should be noted that many websites have taken to sharing their database contents to some extent through a web *a*pplication *p*rogramming *i*nterface or web API. These web APIs generally provide the results of some predefined database queries over a RESTful[1] or semi-RESTful http interface. And these result sets are provided in some convenient machine readable format (most often some ad-hoc e*x*tensible *m*arkup *l*anguage (XML) format or *j*ava*s*cript *o*bject *n*otation (JSON)). Such APIs solve part of our problem - we now have data formatted for computer consumption and we are one step closer to answering our jazz-pianist-deaths question. However, it is highly unlikely that our rather obscure and esoteric query is covered in the set of predefined database queries that make up the web API. We must resort to crawling the API and making thousands of *h*yper *t*ext *t*ransport *p*rotocol (http) requests to answer our query. Writing a computer program to do this is easy - assuming we know what set of API calls to make - the problem is that this program will take a long time to execute. Each http request takes some time to complete as any casual web user knows. Furthermore the host of the web API would likely place some restrictions on how many http requests we can make in a given time window.

The difficulty we find in programming a computer to use the web to answer questions like our jazz-pianist-deaths question has motivated the development of an array of technologies intended to realize the vision of the semantic web [Berners-Lee and Fischetti, 1999; Berners-Lee et al., 2001]. Tim Berners-Lee, the inventor of the web, writes:

> "The Semantic web is not a separate web but an extension of the current one, in which information is given well-defined meaning,

[1] the details of the REST (*RE*presentational *S*tate *T*ransfer) architecture are described by Fielding [2000]

better enabling computers and people to work in cooperation."

In this chapter we will discuss some of these technologies and how they can be applied to music-related data on the web. In section 4.1 we discuss the URI concept - the backbone of the web - and how its role is extended in the semantic web. In section 4.2 we discuss the Resource Description Framework - the data modeling approach at center of Semantic web technologies. In section 4.3 we discuss the role of ontologies in the Semantic web and section 4.4 we discuss the Music Ontology. In section 4.6 we discuss the linked data approach to the semantic web and how it has been a boon for music-related data on the web. We discuss the *S*PARQL *p*rotocol and *R*DF *q*uery *l*anguage in section 4.5 and show how it can be used to answer our jazz-pianist-deaths query. Finally in section 4.8 we discuss some limitations of current semantic web technologies.

4.1 The Uniform Resource Identifier

The Uniform Resource Identifier or URI provides the foundation for the web. The URI is a superset of the more familiar Uniform Resource Locator or URL. As the name implies, URIs are simply strings meant to identify resources. Most commonly URIs relate to the hyper text transfer protocol (http) and are prefixed with "http://" although other prefixes can be used as well (for example the file transfer protocol prefix "ftp://" is also common).

In their most intuitive form, URIs refer to things that are available on the web and can be retrieved directly via http (e.g. html documents). These things are *information resources*. However, URIs and http URIs in particular can be used to refer to things that can not literally be retrieved via http such as people, places, or events. These things are called *non-information resources*. We will revisit this distinction when we discuss *linked data* in section 4.6 but for now it is sufficient to understand that a URI is a unique web name for some thing or concept that may or may not actually be part of the web. This somewhat liberal usage of the URI enables some interesting features as we will see as we continue our discussion of Semantic web technologies. First we will discuss how URIs are used in the Resource Description Framework.

4.2 The Resource Description Framework

The Resource Description Framework or RDF is simply a language for representing information about resources. Because of its well-specified design and incredible level of extensibility RDF has been a W3C recommendation since 1999 [Lassila and Swick, 1999; Beckett, 2004a].

In RDF data is encoded as *triples*. These triples have the form of *subject*, *property*, *object*. Most commonly each element of the triple is a URI or a literal (a string, number, etc). However it is often convenient to have unnamed or *blank nodes* in an RDF graph. A collection of triples results in a directed labeled graph where the nodes are subjects and objects and the edges are properties. In this sense RDF shares the same graph theoretic model we applied for our artist network analysis in chapter 3.

These concepts are perhaps best illustrated with some simple examples. Let us suppose we want to encode the following statement in RDF:

> There is a *person* identified by `http://dbpedia.org/resource/Bill_Evans` whose name is *Bill Evans* and who *plays* the *piano*.

We will encode this statement as a series of triples using URIs for each element as shown in listing 4.1:

```
<http://dbpedia.org/resource/Bill_Evans>
  <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    <http://xmlns.com/foaf/0.1/Person> .
<http://dbpedia.org/resource/Bill_Evans>
  <http://xmlns.com/foaf/0.1/name>
    "Bill Evans" .
<http://dbpedia.org/resource/Bill_Evans>
  <http://dbpedia.org/property/instrument>
    <http://dbpedia.org/resource/Piano> .
```

Listing 4.1: An NTriples listing of an RDF graph encoding that a person identified by `http://dbpedia.org/resource/Bill_Evans` has the name is Bill Evans and also plays the piano.

In listing 4.1 each triple consists of three URIs in angle brackets terminated by a period. The exception is the second triple in our listing which consists of two URIs for the subject and property and then a literal "Bill

Evans" as the object. This listing is similar to the NTriples serialization format for RDF[2] although strictly speaking in NTriples one triple should appear on one line - here each triple is broken up into three lines to more clearly fit on the printed page.

We can see that the above statement was encoded using three triples. It is often helpful when thinking about RDF graphs to translate these triples into a sort-of "caveman English" where, as humans, we mentally parse some pronounceable fragment of each element of the triple and string these fragments together into a broken sentence. From listing 4.1 we might mentally create a sentence like, "Bill_Evans type Person. Bill_Evans name Bill Evans. Bill_Evans instrument Piano." Hopefully this makes it clear that we've been successful in encoding our target set of facts. What is probably not clear is exactly *where* all these URIs came from and *why* they have any meaning. The origins of these URIs will become clear as we discuss ontologies in section 4.3 and linked data in section 4.6. For now just understand that we are using URIs from the DBpedia project which is discussed in section 4.6.2.

Note that we can also visualize this RDF segment as a directed labeled graph as shown in figure 4.1.

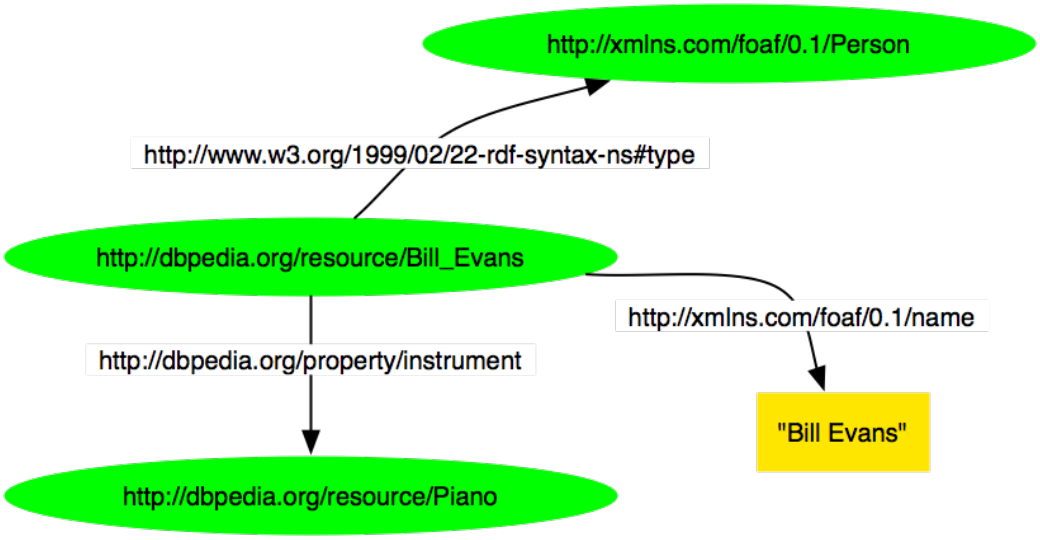


Figure 4.1: An RDF graph encoding that a person identified by `http://dbpedia.org/resource/Bill_Evans` has the name is Bill Evans and also plays the piano.

In figure 4.1 we see the same three triples expressed in 4.1. We can imagine

[2] `http://www.w3.org/TR/rdf-testcases/#ntriples`

encoding additional facts and expanding this RDF graph with additional nodes and edges. Of course the graph would become very large rather quickly so the approaches for serializing RDF graphs or visualizing the graphs we've seen so far would become rather cumbersome. For this reason we will be working with alternative RDF syntaxes.

4.2.1 RDF Syntaxes

There are several ways to serialize RDF. The RDF/XML serialization is based on the extensible markup language (XML) and is described in section 4.2.2. All other RDF serializations are based on some subset of Notation 3 (N3). Turtle described in section 4.2.3 provides expressiveness equivalent to the RDF model while N3 described in section 4.2.4 provides some extensions to the RDF model.

4.2.2 RDF/XML

The RDF/XML syntax is a W3C recommendation that is based on XML syntax and provides a widely-supported interchange format for RDF. We can encode the same RDF graph about Bill Evans in RDF/XML as follows:

```
<?xml version="1.0"?>
<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:foaf="http://xmlns.com/foaf/0.1/"
 xmlns:property="http://dbpedia.org/property/">
 <foaf:Person rdf:about="http://dbpedia.org/resource/Bill_Evans">
  <foaf:name>Bill Evans</foaf:name>
  <property:instrument rdf:resource="http://dbpedia.org/resource/Piano"/>
 </foaf:Person>
</rdf:RDF>
```

Listing 4.2: An RDF/XML encoding of an RDF graph encoding that a person identified by `http://dbpedia.org/page/Bill_Evans` has the name is Bill Evans and also plays the piano.

In listing 4.2 we have encoded the same three triples we introduced in listing 4.1. The two RDF graphs are identical, only the serialization syntax is different. In listing 4.2 we use the XML namespace directive to introduce some prefixes. For example the `xmlns:foaf="http://xmlns.com/foaf/0.1/"` directive allows us to use the prefix "foaf:" as an abbreviation. We will discuss prefixes more as we discuss other serialization formats.

While RDF/XML is widely-supported and easily parsed by machines it can be a bit verbose and not very human-readable. For this reason we will only provide this mention of RDF/XML and focus on other serialization options. Additional details about RDF/XML can be found in W3C RDF/XML Syntax Specification Document [Beckett, 2004b].

4.2.3 Turtle

Turtle or Terse RDF Triple Language is an RDF serialization format that is more human-readable than RDF/XML. It is a subset of the N3 language (which is described in section 4.2.4) and matches exactly the expressiveness of the RDF model. As of this writing turtle is a W3C team submission awaiting recommendation status although it is already widely supported by semantic web applications and libraries [Beckett and Berners-Lee, 2008].

Let us encode the same RDF graph about Bill Evans in Turtle:

```
@base <http://dbpedia.org/resource/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix property: <http://dbpedia.org/property/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

:Bill_Evans rdf:type foaf:Person ;
  foaf:name "Bill Evans" ;
  property:instrument :Piano .
```

Listing 4.3: An Turtle encoding of an RDF graph encoding that a person identified by `http://dbpedia.org/page/Bill_Evans` has the name is Bill Evans and also plays the piano

Here the `@base` directive is used to abbreviate our data URIs replacing `http://dbpedia.org/resource/` with `:`. If the `@base` is omitted, the `:` symbol refers to the context of the current document. Similarly the `@prefix` directive allows us to abbreviate additional URI namespaces. By convention the `@prefix` directive is used for vocabulary or ontology namespaces. When we use an abbreviation defined by the `@base` or `@prefix` directive we are creating a qualified name or qname. These qnames are distinguished from unabbreviated URIs by their lack of enclosing angle brackets. For the sake of brevity and clarity we will assume a number of prefixes in the listings throughout the remainder of this thesis. These assumed prefixes are listed in appendix A.

As in the NTriples syntax in listing 4.1, triples in the Turtle syntax consist of three elements terminated by a period. However, the semicolon character "`;`" can be used to group triples that share the same subject. This is done in listing 4.3. Similarly the "`,`" character can be used to group triples that share both the same subject and same property element.

Also note the special property `rdf:type`. This property is used to specify instances of a class - something we will discuss more in section 4.3. In the Turtle syntax the `rdf:type` property can be and often is abbreviated simply as "`a`".

Blank nodes can be expressed in Turtle as $\_:x$ where x is some arbitrary identifier used in the serialization. Alternatively, blank nodes can be expressed as a list of property object pairs applying to that blank node enclosed in square brackets. Additional details of the RDF Turtle syntax are available in [Beckett and Berners-Lee, 2008].

4.2.4 Notation 3

Notation 3 is both an RDF syntax and an extension of the RDF model. As mentioned before the syntax for N3 is a superset of the Turtle syntax (Turtle was actually developed after N3 as a subset of N3 that matches exactly the expressiveness of the RDF model). N3 includes several additional facets including paths, additional keywords, and the ability to quote formulæ [Berners-Lee, 1998; Berners-Lee et al., 2007]. We may treat a list of triples - a formula - as a literal value or a *graph literal*. This effectively allows us to nest RDF graphs and enables us to make statements about an entire RDF graph. This approach is similar to the named graphs approach described in [Carroll et al., 2005]. The primary difference is that in N3 formulæ quoting, the name of the graph is its value rather than an additional minted URI.

In N3 quoting is accomplished by enclosing a set of triples inside curly brackets. An example is given in listing 4.4:

```
{ [ foaf:name "Bill Evans" ] foaf:made [ dc:title "Blue in Green"] }
        a log:Truth .
```

Listing 4.4: An N3 encoding of the statement "it is true that there is someone named Bill Evans who made something titled Blue in Green."

This listing states, "It is true that there is someone named Bill Evans who made something titled Blue in Green." As in turtle, web identifiers are

either between angle brackets or in a `prefix:name` qname notation. The square brackets denote an existentially quantified entity or blank node. The contents between are predicate object pairs that apply to the blank node (note this syntax is also allowed in turtle). Curly brackets denote a literal resource corresponding to a particular RDF graph. This is an extension to turtle available in N3 that allows us to make statements about a collection of triples.

Another extension of the RDF model provided by N3 is that of the universally quantified variable. Universal quantification formalizes the notion that something is true for every relevant value of the variable. This is contrary to existential quantification which is the predication of a property or relation to at least one value for the variable. Existential quantification is provided in both RDF and N3 and is modeled using the blank node concept described in section 4.2.3. In summary universal quantification relates to the notion of "for all" while existential quantification relates to the notion of "for some". In the N3 syntax the keywords `@forAll` and `@forSome` can be used to specify universal quantification and existential quantification respectively. However, it is more common to simply prefix a universally quantified variable with `?` (e.g. `?x`) and an existentially quantified variable with the blank node notation `_:` (e.g. `_:x`). The use of an universally quantified variable is demonstrated in listing 4.5:

```
?x a mo:SoloMusicArtist, foaf:Person .
```

Listing 4.5: An N3 encoding of the statement "all values of x are both solo music artists and persons."

This simply expresses that all values of `?x` are both solo music artists and persons. Additional details about the N3 syntax are available at [Berners-Lee et al., 2007; Berners-Lee, 1998].

4.2.5 Other RDF Syntaxes

There are additional syntaxes for the serialization of RDF. In listing 4.1 we showed the basic aspects of the NTriples[3] format. NTriples is a subset of the Turtle syntax which does not make use of prefixes or other abbreviations. The RDFa[4] syntax allows the embedding of RDF data within XHTML documents

[3] see http://www.w3.org/TR/rdf-testcases/#ntriples
[4] see http://rdfa.info

and is becoming an increasingly popular method of integrating semnatic web technologies into existing websites. Details of the syntax are not provided here as we do not make use of the RDFa syntax in this work.

This is not to discount the importance of this technology. In allowing developers to embed the rich semantic information of RDF directly into existing HTML web pages, RDFa is probably one of the most import semantic web technologies from the perspective of fostering uptake. Note that the recent OpenGraph protocol[5] that allows rich content integration between 3rd-party websites and the FaceBook platform is based on RDFa.

4.3 Ontologies

We have seen how RDF and URIs can be used to encode data. But how is meaning ascribed to those URIs? How does this data become useful information or knowledge? The answer is through the use of *ontologies*. Semantic web ontologies ascribe meaning to URIs via relational semantics. These ontologies define classes and properties. Resources are declared instances of classes and appropriate properties are associated with those instances as triples. In this sense the semantic web is essentially an attempt at building a distributed web-scale knowledge representation framework where agents can reason against the knowledge of the web.

4.3.1 RDF Schema

The RDF Schema or RDFS is a vocabulary for describing classes and properties related to various resources in RDF [Brickley and Guha, 2004]. RDFS is itself authored in RDF demonstrating the extensibility of the RDF model. A given ontology will generally consist of a set of `rdfs:Class` instances and `rdfs:Property` instances. Then related RDF data which makes use of this ontology will consist of instances of the `rdfs:Class` instances specified in the ontology.

For example we might extend the RDF model in listing 4.3 with some ontological information as follows:

RDFS also provides specifying class hierarchies through the `rdfs:subClassOf` property. Declaring hierarchies of classes can enable inferencing through subsumption. For example, if we know that We can then infer that the prop-

[5] see http://developers.facebook.com/docs/opengraph

```
foaf:Person a rdfs:Class .
:Bill_Evans a foaf:Person .
```

Listing 4.6: A turtle listing defining a class for Person and `:Bill_Evans` as an instance of that class.

```
mo:MusicArtist rdfs:subClassOf foaf:Person .
```

Listing 4.7: A turtle listing defining `mo:MusicArtist` as a subclass of `foaf:Person`

erties of `foaf:Person` also apply to `mo:MusicArtist`. We know that since a `foaf:Person` has a `foaf:birthday`, a `mo:MusicArtist` must also have a birthday.

RDFS also allows us to scope a property by explicitly specifying which class instances can be the subject of the given property (`rdfs:domain`) and which class instances can be the object of the given property (`rdfs:range`).

Additional details about RDFS can be found in the W3C Recommendation [Brickley and Guha, 2004].

4.3.2 The Web Ontology Language

The Web Ontology Language or OWL is another W3C Recommendation that relates to the creation of ontologies and knowledge representation. OWL is actually a family of languages based on two largely but not entirely compatible semantics: the OWL DL/Lite semantics are based on Description Logic and have well-understood computational properties while OWL Full is based on a semantic model that provides compatibility with RDFS.

In particular OWL allows modelers to express detailed constraints between classes, entities, and properties. For example we could use OWL to add some restrictions to a particular ontological definition.

In listing 4.8 we specify an `owl:equivalentClass` to the `mo:Conductor` class that imposes an `owl:Restriction` (which is modeled as a blank node here). This restriction uses the `owl:someValuesFrom` property to specify that a `mo:Conductor` must have `mo:conducted` at least one `mo:Performance`.

OWL also provides full facilities for describing set theory including intersections, unions, and complements. For example a new class can be defined

```
mo:Conductor owl:equivalentClass [
        a owl:Restriction;
        owl:onProperty mo:conducted;
        owl:someValuesFrom mo:Performance
    ] .
```

Listing 4.8: Adding a restriction to the conductor class specifying that a conductor must have conducted a performance.

as a union or intersection of previously defined classes.

The OWL vocabulary also provides a simple but important facility for stating that two resources are actually one-in-the-same. The `owl:sameAs` predicate can be used to make such statements. For example we might use this predicate to indicate a music artist identified by a DBPedia URI is the same as a music artist identified by a DBTune URI. In this way we can reconcile information from multiple sources which use distinct URIs to identify the same thing. This also enables the most basic, but perhaps most important, inferencing. A reasoning engine can infer that triples pertaining to the subject of an `owl:sameAs` predicate also apply to the object and vice versa.

Additional details about OWL are available in the W3C recommendation document [Bechhofer et al., 2004].

4.4 The Music Ontology

As the name implies, the Music Ontology[6] is a web ontology for describing resources in the music domain. By focusing on temporal annotations and even decomposition the Music Ontology provides what is arguably one of the most expressive and complete approaches to encoding music-related knowledge [Raimond et al., 2007; Abdallah et al., 2006; Raimond, 2009].

In the music domain there is a variety of complex scenarios that require a flexible modeling framework. Consider the composition "Caravan" - a jazz standard composed by Juan Tizol and first popularized by Duke Ellington. Countless musical acts have since performed their own versions of this song and Ellington himself recorded over 80 versions. Some of these versions were recorded live some were recorded in the studio. To add even more complexity,

[6] specification available at `http://musicontology.com`

rap artist Redman later sampled an Ellington recording of "Caravan" in his 1998 song "Da Goodness". The artist name, track name, and album title fields often found attached as metadata for digital audio files only encode a small fraction of the information related to our Caravan scenario. With the Music Ontology we can embrace the complexity of this scenario and encode much of the knowledge related to this particular composition using a flexible and extensible RDF model.

To model such complexity the Music Ontology combines the layered abstraction approach outlined in the Functional Requirements for Bibliographic Records (FRBR) [on the Functional Requirements for Bibliographic Records, 1998; Davis and Newman, 2005] with a modeling mechanism for describing timelines and events.

4.4.1 Functional Requirements for Bibliographic Records

Four layers of abstraction are described in FRBR. These include:

- *Work* – A distinct intellectual creation;
- *Expression* – An artistic realisation, for example the product of a musical performance;
- *Manifestation* – A group of similar items embodying an expression, for example an album;
- *Item* – A single exemplar of a manifestation, for example an audio file or a copy of an album.

FRBR helps us to capture additional details of our Caravan scenario. We can model "Caravan" the composition as an FRBR Work and the various performances of "Caravan" can be modeled as FRBR Expressions. However, we are still unable to model the complexity of the relationship between Redman's "Da Goodness" and "Caravan". FRBR alone does not allow us to model which *segment* of which recording of "Caravan" was used in "Da Goodness".

4.4.2 Timeline Ontology

To effectively model the music domain the Music Ontology must deal with the temporal aspects of music. The OWL-Time ontology provides some formal

modeling of time on the semantic web [Hobbs and Pan, 2004; Pan, 2007]. However, in the music domain it is necessary to consider multiple timelines (e.g. the timeline corresponding to the recording of "Da Goodness" and the timeline corresponding to a particular the recording of "Caravan"). For this reason Raimond et al. extend OWL-Time with the Timeline Ontology[7] [Raimond and Abdallah, 2006b].

The Timeline Ontology adds concepts like `tl:TimeLine` and `tl:TimeLineMap` which not only allow us to model multiple timelines but also model the mapping between two timelines. Additional Detail of the Timeline Ontology are available in [Raimond and Abdallah, 2006b; Raimond, 2009].

4.4.3 Event Ontology

In addition to specifying timelines, we want to specify regions on timelines. Revisiting our Caravan scenario we might want to specify, "the saxophone solo on this particular recording of Caravan appears here." The Event Ontology [Raimond and Abdallah, 2006a] defines event tokens which can then be associated with a given timeline.

4.4.4 Caravan Modeling

With the FRBR abstraction layering and the event decomposition capabilities we can use the Music Ontology to model the Caravan scenario mentioned at the beginning of this section.

In listing 4.9 we provide a Turtle RDF model describing the composition of Caravan, a particular recording of Caravan by Duke Ellington, and the subsequent sampling of that recording by Redman.

Note we do not specify an `@base` parameter so our qnames beginning simply with the prefix `:` are defined only in the context of the listing. We specify a composition event with the blank node `_:Composition_of_Caravan` that produced the work identified as `dbpedia:Caravan_(song)`. We then have a performance of that work identified as `:Ellington_Performance` which is subsequently recorded as a `mo:Signal`. We then use the `mo:sampled` property to relate the artist `dbpedia:Redman` to the sampled signal.

[7] The Timeline Ontology namespace is `http://purl.org/NET/c4dm/timeline.owl#` and an ontology specification document can be retrieved at the same URL

```
dbpedia:Juan_Tizol a mo:MusicArtist.

_:Composition_of_Caravan a mo:Composition ;
  dc:date "1936"^^xsd:Date ;
  mo:composer dbpedia:Juan_Tizol ;
  mo:produced_work dbpedia:Caravan_(song) .

dbpedia:Caravan_(song) a mo:MusicalWork .

:Ellington_Performance a mo:Performance ;
  dc:date "1962"^^xsd:Date ;
  mo:performance_of dbpedia:Caravan_(song) ;
  mo:recorded_as :Signal_of_Ellington_Performance .

:Signal_of_Ellington_Performance a mo:Signal ;
  mo:published_as :Money_Jungle_Caravan .

discogs:release/1852044 a mo:Record ;
  mo:track :Money_Jungle_Caravan ;
  dc:created "1962"^^xsd:Date ;
  dc:Title "Money Jungle" .

dbpedia:Redman mo:sampled :Signal_of_Ellington_Performance .
```

Listing 4.9: A Turtle RDF model describing the composition of Caravan, a particular recording of Caravan by Duke Ellington, and the subsequent sampling of that recording by Redman.

4.4.5 Adoption

There has been some significant adoption of the Music Ontology in the broader web community. The BBC now models information about music artists using the Music Ontology for their website. The Music Ontology has also been used in the RDF translation of MetaWeb's Freebase[8], Talis's Dataincubator[9] projects, and a variety of other projects that are part of the linked data movement we will discuss in section 4.6.

[8] see http://rdf.freebase.com/
[9] see http://dataincubator.org/

4.5 SPARQL an RDF Query Language

We have seen how semantic web technologies can be used to model complex scenarios in music and other domains. But how do we use these models? How can we query the knowledge we've worked so hard to encode? The answer lies in what is arguably the most exciting semantic web technology SPARQL (pronounced "sparkle"). SPARQL is a recursive acronym that stands for SPARQL Protocol and RDF Query Language and it is an official W3C Recommendation [Prud'hommeaux and Seaborne, 2008].

SPARQL allows us to query an RDF graph using triple patterns, conjunctions, disjunctions, and optional patterns. The query syntax is derived from N3 syntax (discussed in section 4.2.4) and allows for the creation of impressively expressive queries. Variables in triple patterns are specified using the same notation for universal quantification that is used in N3 - that is variable names are prefixed with a "`?`". The keywords `SELECT` and `WHERE` to specify the variables of interest and the triple pattern query respectively.

In listing 4.10 we construct a simple query to retrieve a list of music artists and their names from an arbitrary RDF graph.

```
SELECT ?artist ?name WHERE {
  ?artist a mo:MusicArtist;
    foaf:name ?name .
}
```

Listing 4.10: A SPARQL query for retrieving a list music artists and their names

Note that the variables in which we are interested are specified in the `SELECT` clause in this case we are selecting `?artist` and `?name`. Also note these same variables are used in the triple patterns in the `WHERE` clause. We will use SPARQL to answer the jazz-pianst-deaths query in section 4.7.

4.6 Linked Data

Even proponents acknowledge that the realization of the semantic web vision is not complete and that uptake of semantic web technologies has been slower than some expected [Feigenbaum et al., 2007]. The linked data movement has revitalized semantic web technology.

The traditional "top-down" approach of designing an ontology *first* and then developing the data breaks down at the scale of the web. The linked

data approach allows for a more incremental and grass-roots development of the semantic web. This collaborative approach allows for ambiguity and inconsistency while providing data provenance and implicit knowledge in a uniform manner. In some respects, the linked data movement represents a shift in focus with respect to the semantic web - away from reasoning and inference and towards data modeling and practical utility. Although this is not strictly the case. Many linked data projects have a grounding in description logics and inferencing.

At the most basic level, linked data simply outlines a set of best practices for publishing data on the semantic web [Berners-Lee, 2006; Bizer et al., 2008]. Linked data URIs must dereference via http and provide useful information in an appropriate web standards-based format. This process usually involves some content negotiation and the details of the http request header. And the dereferenced information should include links to additional http URIs that also dereference fostering the discovery of additional information. The result is a mesh of web-accessible resource descriptions that can span the boundaries of organizations, hardware infrastructure, and domains of discourse.

The Linking Open Data (LOD) project aims to publish open data sets on the semantic web following linked data recommendations and appropriate web standards. A diagram of some of the linked data sets maintained by the LOD community can be seen in figure 4.2. Note the pale blue circles indicate data sets that are related to music.

Many of these music-related LOD datasets are hosted by the DBTune project.

4.6.1 The DBTune Project

The DBTune[10] website has been serving music-related linked data sine early 2007 [Raimond, 2009; Bizer et al., 2007]. As part of the Linking Open Data project DBTune hosts several music-related RDF datasets. These include the Jamendo dataset, the Magnatune dataset, the John Peel sessions dataset, the AudioScrobbler wrapper, a MusicBrainz translation, the BBC playcounts dataset, and Henry. New contributions to the DBTune project made as part of our work include the MySpace wrapper, the last.fm artist similarity service, the Echonest artist similarity service, and the Classical Composers dataset.

[10] http://dbtune.org/

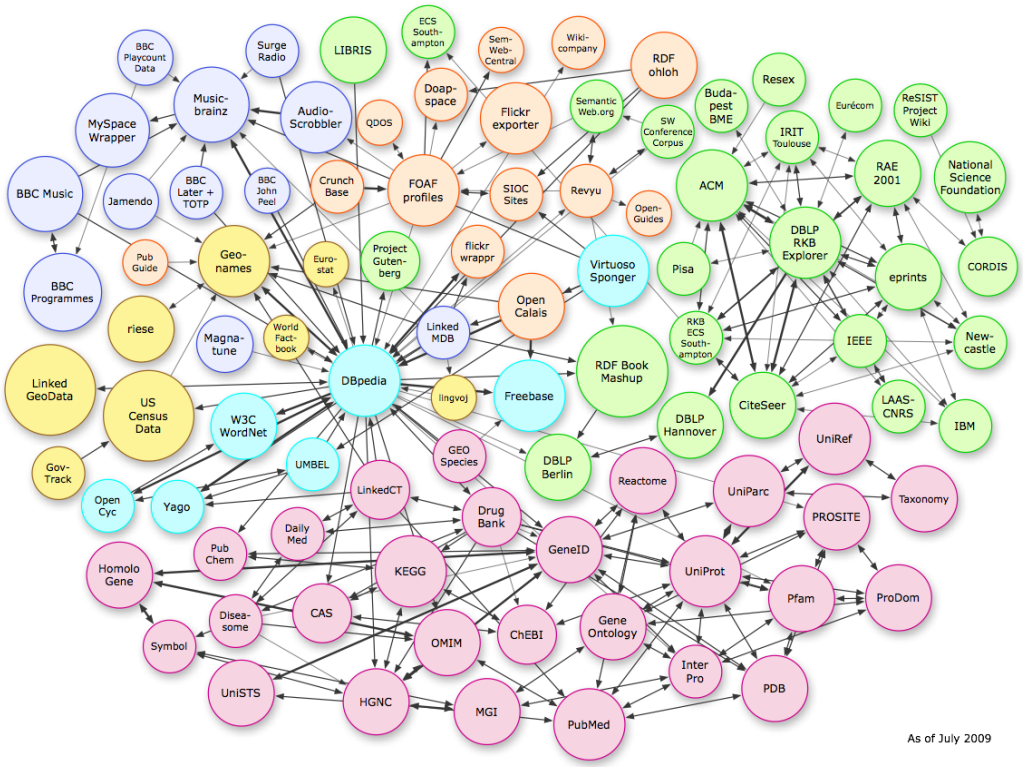


Figure 4.2: Linking Open Data cloud as of November 2009 where the pale blue circles indicate data sets related to music.

Jamendo Dataset

The DBTune Jamendo dataset[11] includes information about Jamendo music artists, their geographic locations, and their works expressed in RDF using Music Ontology concepts and appropriate concepts from other ontologies. Jamendo [12] is a website that enables music artists to publish music under a creative commons license that allows listeners to download and share music legally.

Magnatune Dataset

The DBTune Magnatune dataset[13] includes information about Magnatune music artists and their works expressed in RDF using Music Ontology con-

[11] see http://dbtune.org/jamendo/
[12] see http://www.jamendo.com/
[13] see http://dbtune.org/magnatune/

cepts and appropriate concepts form other ontologies. Like Jamendo, Magnatune[14] enables music artists to publish their works under a creative commons license. While Jamendo is more of an open platform, Magnatune is more like a traditional record label that hand-picks artists and content for publication.

John Peel Sessions

The DBTune John Peel sessions dataset[15] describes metadata related to the various recordings associated with the long-running John Peel BBC 1 radio show in RDF.

AudioScrobbler Wrapper

The DBTune AudioScrobbler wrapper[16] provides information for a given last.fm user about the last 10 played tracks as RDF linked to DBTune MusicBrainz dereferencable URIs.

MusicBrainz Dataset

The DBTune MusicBrainz dataset[17] provides an RDF mapping of the extensive music metadata database maintained by the MusicBrainz community (discussed in section 3.9). As described in section 3.9.1 MusicBrainz provides unique identifiers for music-related data including music artists, tracks, and albums making it an ideal resource for creating linked data URIs for music-related entities. A dump from the MusicBrainz Postgres database is used to power a D2R server[18], which, given the appropriate mappings, provides the data in the relational database as RDF and makes available a SPARQL end-point.

BBC Playcount Dataset

The BBC playcount dataset[19] contains information about which music artists are played on which BBC Programmes as RDF with links to the DBTune

[14] see http://magnatune.com/
[15] see http://dbtune.org/bbc/peel/
[16] see http://dbtune.org/last-fm/
[17] see http://dbtune.org/musicbrainz/
[18] see http://www4.wiwiss.fu-berlin.de/bizer/d2r-server/
[19] see http://dbtune.org/bbc/playcount/

Musicbrainz data set.

Henry

The DBTune Henry server is a workflow processor for audio processing tasks that builds on the expressiveness of Notation 3 (see section 4.2.4) with the state-changing awareness of transaction logic. We will discuss Henry and the associated N3-Tr syntax in more detail in section 5.2.5.

New Contributions

As part of the work supporting this thesis we have created or helped to create the following extensions to the DBTune project:

- The MySpace wrapper[20] provides URIs and associated RDF representations for top-friends, country of origin and available tracks by scraping MySpace web pages. The results of page scrapings are also cached in a triple store to enable a SPARQL endpoint.
- The last.fm artists wrapper[21] uses the MuSim ontological framework we develop in chapter 5 to describe inter-artist similarities from the last.fm API and includes inter-linking to the DBTune/MusicBrainz data set and BBC/Music.
- The Echonest artists wrapper[22] uses the MuSim ontological framework developed in chapter 5 to describe inter-artist similarities from the Echonest API and includes inter-linking to the DBTune/MusicBrainz data set and BBC/Music.
- The Classical Composer data set[23] provides an array of information about concepts and individuals related to the canon of Western Classical Music. This includes the composer influence relations from the Classical Music Navigator discussed in section 3.4 as well as data aggregated from around the web. This data set is, to some extent, hand curated by Chris Cannam and provides appropriate links to resources in DBpedia, DBTune/MusicBrainz, and BBC/Music.

[20] see http://dbtune.org/myspace/
[21] see http://dbtune.org/artists/last-fm
[22] see http://dbtune.org/artists/echonet/
[23] see http://dbtune.org/classical/

4.6.2 The DBPedia Project

The DBPedia[24] project provides a rich mapping of concepts and topics from Wikipedia into RDF. DBpedia provides a rich corpus of diverse data by leveraging the crowd-sourced "web 2.0" content of Wikipedia with semantic web technologies. The DBpedia project provides a extensible information extraction framework that operates on the Wikipedia database dumps. Semantic relationships are extracted from tables in the relational database and directly from the article texts, infobox templates and categorization information. The DBpedia data set currently provides information about more than 1.95 million "things" including at least 80,000 persons, 70,000 places, 35,000 music albums, 12,000 films and many other topics [Auer et al., 2007].

The DBPedia resources serve as the defacto hub for dataset interlinking as is apparent from Linking Open Data diagram in figure 4.2. The breadth of the DBpedia dataset makes it an easy target for interlinking and the depth of the information makes it an attractive target as well. In our work, we provide links to the DBpedia dataset whenever possible.

Of course many music-related topics are covered in Wikipedia and hence encoded as RDF in DBPedia. We will show how we can use the data in DBPedia to answer our jazz-pianist-deaths query as we discuss SPARQL in the section 4.7.

4.7 Answers from Linked Data and SPARQL

Recall our jazz-pianist-deaths query from the beginning of this chapter: "Show me a list of deceased jazz piano players and their respective causes of death". With SPARQL, we can use the RDF graphs in DBPedia (described in section 4.6.2) to answer this query. DBPedia provides a *SPARQL Endpoint*[25] which allows an arbitrary agent to make SPARQL queries against the DBPedia RDF. We can get some results for our jazz-pianist-deaths query from the DBPedia endpoint with the following SPARQL query:

DBPedia uses the Simple Knowledge Organizational Scheme (SKOS) ontology to encode the heirarchical structure of the Wikipedia categories into RDF. We use the `dbpedia:Category:Deaths_by_cause`, which is an instance of `skos:Concept`, as our top concept and retrieve a variety of subconcepts using the `skos:broader` property. However, SPARQL, as of this writing,

[24] http://dbpedia.org/

[25] the DBPedia SPARQL endpoint is available at http://dbpedia.org/sparql

```
SELECT DISTINCT ?name ?cause_of_death ?date_of_death
WHERE {
?artist skos:subject ?cause_of_death ;
   foaf:name ?name ;
   dbpedia2:instrument dbpedia:Piano ;
   dbpedia-owl:genre dbpedia:Jazz ;
   dbpedia-owl:deathDate ?date_of_death .
?cause_of_death skos:broader ?d .
{ ?d skos:broader dbpedia:Category:Deaths_by_cause . }
UNION
{ ?d skos:broader ?dd .
  ?dd skos:broader dbpedia:Category:Deaths_by_cause . }
UNION
{ ?d skos:broader ?dd . ?dd skos:broader ?ddd.
  ?ddd skos:broader dbpedia:Category:Deaths_by_cause . }
UNION
{ ?d skos:broader ?dd . ?dd skos:broader ?ddd.
  ?ddd skos:broader ?dddd.
  ?dddd skos:broader dbpedia:Category:Deaths_by_cause . }
}
```

Listing 4.11: A SPARQL query against DBPedia for retrieving a list of dead jazz pianists and their respects causes of death

does not support transitive closure. That is, we can not accommodate hierarchies of an arbitrary length. Some extensions to the SPARQL do support transitive paths but for our example we use a series of `UNION` statements to explicitly traverse a transitive path with four steps. This brings us to some of the limitations of current semantic web technologies.

4.8 Limitations of Current Semantic Web Technology

As mentioned in the previous section, SPARQL currently has no support for transitive closure. This turns out to be a rather significant limitation. Consider the music artist networks we discussed in chapter 3. Suppose we have modeled inter-artist connections using a simple property such as `foaf:knows`. Given a specific artist, it is simple enough to find the list of artists that are immediately adjacent in the network. But suppose we want to continue to find *all* the artists that belong to this connected component. We can create a set of `UNION` clauses as we did in listing 4.11 or make repeated queries to

achieve a fixed depth. In addition to being verbose, a bit ugly, and potentially very inefficient, this approach forces us to specify a specific fixed depth - if we do not know anything about the network a priori it becomes quite complicated to deal with arbitrary path lengths. The sort of analysis we present in chapter 3 is possible with semantic technologies but not trivial.

Some solutions to the transitive closure problem have been suggested. In an environment which supports querying over an inferred graph, inference rules can be used to specify transitive closures or hierarchy membership relations that can then be queried with SPARQL. Some such inference engines exist, for example the Closed World Machine[26] allows the parsing of full N3 and supports some "magic predicates" which trigger the inference of additional triples. Also, the Virtuoso software supports a variety of extensions to SPARQL[27] that allow an inference context for inferring triples that are not explicitly stored in the RDF store.

Additional difficulties found in applying semantic web technologies to the domain of music have been discussed by [Cannam et al., 2010]. For example, RDF data is often distributed between several different SPARQL endpoints. While distinct data sets may (or may not) contain RDF that uses compatible ontologies and URI schemes, querying across multiple endpoints with one federated query is currently not well-supported. The current SPARQL recommendation does not support federated queries, and there are no standard means of discovering, pooling, and caching multiple documents about a subject. However the pending SPARQL 1.1 working draft[28] includes a specification for federated queries.

Simple RDF literals are sometimes encoded using inconsistent data types. This is problematic because `''67''` is not equivalent to `''67''xsd:int` in the SPARQL specification. This makes queries hard to write and, more seriously, unreliable when used with real-world data from multiple sources.

Furthermore, no efficient means of structuring numerical data such as vectors or matrices exists in RDF. This becomes a problem when dealing with dense numerical data such as audio-based analysis features. The Opaque Features Files Ontology[29] has been created to partially address this issue, however this project is incomplete as of this writing.

There are also some very basic issues inherent to semantic web technologies that are worth mentioning. The process of ontology creation and on-

[26] see http://www.w3.org/2000/10/swap/doc/cwm.html
[27] see http://docs.openlinksw.com/virtuoso/rdfsparqlrule.html
[28] see http://www.w3.org/TR/2010/WD-sparql11-federated-query-20100601/
[29] see http://purl.org/ontology/off/

tology maintenance requires considerable time and effort [Hepp, 2007; Bizer et al., 2009; Vrandečić, 2009]. In some scenarios it is not clear that this effort is warranted. For example, the dynamics of a given domain might require a dynamic ontology - one that evolves and adapts over time. This introduces a variety of design challenges and requires an engineer or team on engineers to constantly update a model while ensuring backwards compatibility or adjusting datasets to deal with a lack-there-of.

The gap between the ontology creation community and the (intended) ontology user community is another concern. An ontology can be very complete and expressive in terms of ontological design principles, but if the user community is unable to leverage this ontology due to a lack of documentation or even disparate views of the domain in question; opportunities for ontology re-use can be lost. This may result in several disparate ontologies describing the same domain [Hepp, 2007]. It can be argued the ability to accommodate contrasting models of the same domain is a strength of semantic web technologies. However in practice disparate models make data fusion an issue (something semantic web technologies are meant to alleviate) and gives rise to the problem of ontology alignment [Stoilos et al., 2005; Euzenat and Valtchev, 2004].

4.9 Summary

In this chapter we have discussed semantic web technologies and how they have been applied to music-related information. Despite some of the difficulties we mention in section 4.8, the utility of semantic web technologies for modeling the complexity of the music domain is unparalleled as demonstrated by the success of the DBTune project and adoption of the Music Ontology. In chapter 5 we will discuss the subtleties of musical associations and how the semantic web technologies we discussed in this chapter can be used to model connections between music artists, compositions, and other entities.

Chapter 5

A Framework for Associations

> *Some things share a complicated network of similarities overlapping and criss-crossing: sometimes overall similarities, sometimes similarities of detail.*
>
> – Ludwig Wittgenstein, *Philosophical Investigations*, 1953

In chapter 3 we saw the wide variety of music-related connections that can be found on the web - focusing on music artist networks. In chapter 4 we saw how semantic web technologies can be used to provide a rich and thorough modeling of music-related knowledge on the web. Now we will propose a distributed framework based on semantic web technologies for modeling the wide variety of music-related connections that can be found on the web and in the real world with an emphasis on transparency and provenance. At the heart of this framework is the Similarity Ontology [Jacobson, 2009] which is often referred to as MuSim - a concatenation of *mu*sic and *s*imilarity. But before developing our ontological framework let us discuss the nature of associations, similarity, and, in particular, music similarity.

5.1 On Similarity

At an intuitive level, similarity is a very simple concept. We all have some general sense about what it means for two things to be similar. But at an analytical level it becomes more difficult. Judgments of similarity are so fundamental to our cognitive process that a concrete definition of the term becomes somewhat elusive and even circular. So let us briefly develop our own definition for the context of this thesis.

In music informatics we are interested in music-related entities - primarily music artists, songs, and audio signals although we are also potentially interested in albums, record labels, segments of songs, chords, melodies, instrumentations, listeners, and other objects in the vast variety of concepts related to the music domain. Many of these concepts are handled quite well by the event decomposition approach of the Music Ontology (see section 4.4). However we are particularly interested in the connections between these music-related entities. For some of these inter-concept connections, the Music Ontology and other existing ontologies provide a clear modeling methodology. For example, with the Music Ontology we can make explicit the connections between a composition, a performance of that composition, and a recording of a performance of a given composition (see the Caravan example in section 4.4.4). However we are often interested in connections that are more nebulous and beyond the scope of current ontological modeling. For example we might want to model the similarity between two music artists based on a particular algorithm that analyzed specific audio signals selected from the respective artists' catalogs of songs - similar to the analysis we discussed in section 3.6.4.

We are interested in these extraordinary inter-entity connections for two fundamental reasons: (1) to obtain a better understanding of the cultural and contextual aspects of music and how music is evolving on the web and (2) to facilitate the navigation of the ever-expanding universe of music. We can consider music recommendation as the canonical application of the later. For example to navigate a collection of music artists, we want to be recommended new music artists based on a list of artists we like. We argue that similarity is factotum to recommendation - we want to be recommended items that are somehow similar to the items we like. It is assumed the similar items will share some of the traits that the user finds appealing. This brings us to the definition of similarity we will use in this work and in our framework:

> *Entities share a **similarity** if they share some common characteristics of interest.*

As a broader concept that encompasses similarity we define *association* as follows:

> *Entities share an **association** if they are somehow inter-connected. A similarity is a special type of association.*

This broader association concept will lend our framework a larger degree of flexibility. Note that in this work, what we say about *association* also holds true for *similarity*.

Our definition for similarity is perhaps most in accord with the featural model of similarity proposed by psychologists such as Amos Tversky [Tversky and Gati, 1982]. We will discuss various psychological models of similarity in our evaluation in chapter 6.

When we consider our definition for similarity it becomes evident that similarity is both multifaceted and context dependent. Two items might share a variety of characteristics making their similarity a complex network of individual similarities. And in different applications different facets of a similarity might be more salient than others. For example two pieces of music might share the same metrical structure and rhythmic feel, while having distinct key signatures and melodies and being composed by artists on opposite sides of the globe. If we are interested in constructing a beat-matched DJ mix of these pieces of music, the fact that they share metrical and rhythmic similarities is probably most important. But if we are composing a playlist representative of the music of a specific region, the rhythmic facet of this compound similarity becomes unimportant to us.

Any pair of entities potentially have an infinite number of distinct associations between them. Our framework must accommodate the compound associations between entities and be capable of expressing the details of these associations. Some connections are grounded in indisputable facts (e.g. "Both James Brown and Bill Withers performed at the 1973 Ziare Soul Power concert") while other connections are grounded in opinion (e.g. "David Bowie and Bob Marley are the two best artists of all time") while still others are the result of some algorithmic recommendation process (e.g. "The White Stripes and The Black Keys are similar artists on last.fm"). In a framework for describing connections, we must provide a means of annotating *who* made an association statement and *why*.

It is these intuitions that motivate our Similarity Ontology. Most search and recommendation applications treat inter-item similarity as a single scalar value in a metric space [Chávez et al., 2001] although this is contrary to some of the prevailing psychological models of similarity [Goldstone and Son, 2005]. In this chapter we develop a Similarity Ontology that embraces the multifaceted complexity of similarity and allows for distinct similarities to be specified for distinct contexts.

5.2 The Similarity Ontology

Because of its decentralized nature, wide deployment base, and robust technological underpinnings we use the RDF/OWL framework [Beckett, 2004a;

Bechhofer et al., 2004; Brickley and Guha, 2004] for defining our Similarity Ontology. This allows us to use the concepts, practices, and resources of linked data as discussed in section 4.6.

5.2.1 Association as a concept

In the Music Ontology, similarity is defined as a property `mo:similar_to`. This allows for very simple similarity statements that involve only one triple as illustrated in figure 5.1.

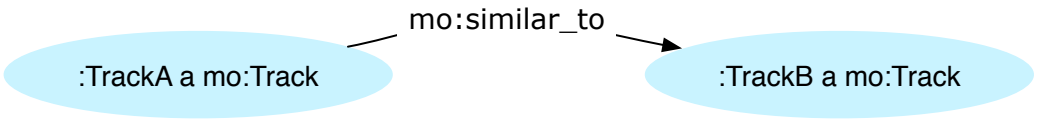


Figure 5.1: Property-based similarity allows us to make similarity statements with one triple

Instead of treating similarity or, to use the broader term, association as a *property*, we treat association as a *class concept*. This allows for easy reification of similarity statements. We introduce the class `sim:Association` and a sub-class `sim:Similarity` as the key concepts in our ontology. We then define the class `sim:AssociationMethod` for describing a method for determining similarity. By associating a similarity statement of type `sim:Similarity` with an instance of `sim:AssociationMethod` we are describing in what sense the elements involved in our statement are similar. We can further reify our method by providing provenance (as discussed in section 5.2.4) and even fully disclose our method by pointing to a graph describing our workflow (as discussed in section 5.2.5). The diagram in figure 5.2 illustrates a basic example involving two music tracks.

Again, we use the top-level concept of `sim:Association` to provide a higher level of abstraction that covers inter-entity connections not necessarily based on similarity. The `sim:Association` concept assumes a definition of association and the concept `sim:Similarity` assumes our definition of similarity, both of which are stated earlier in section 5.1. However, the exact boundary between what should be modeled as a `sim:Association` and what should be modeled as a `sim:Similarity` can be considered application-dependent. For some datasets this distinction might be important be in general the use of the `sim:method` predicate and the `sim:AssociationMethod` modeling paradigm makes these distinctions unimportant.

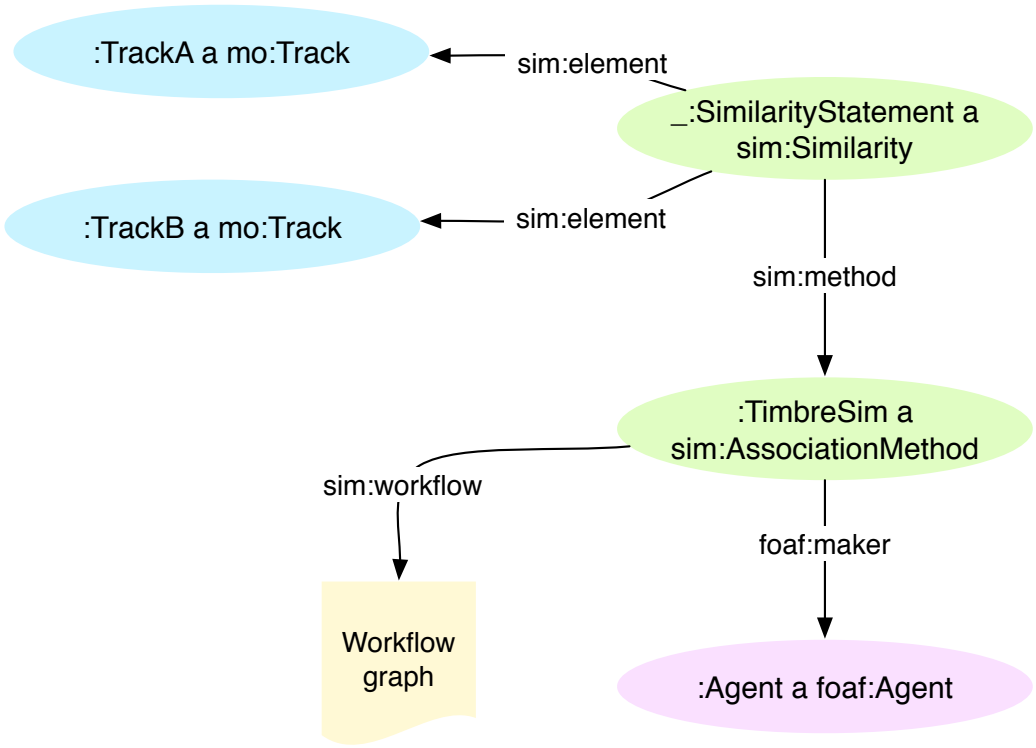


Figure 5.2: A graph visualization of how MuSim concepts are intended to be used. A similarity statement of type `sim:Similarity` is associated with an instance of `sim:AssociationMethod` which in turn leads to more information about the similarity derivation process

Let us begin to develop the details of the Similarity Ontology by examining the very simple similarity example presented in listing 5.1

We introduce the namespace `sim` to refer to our Similarity Ontology (recall a complete listing of the namespaces assumed throughout this work is available in appendix A). First we define two tracks using the corresponding Music Ontology concept `mo:Track`. The identifiers of these tracks can give entry points to additional information in other data sets (i.e. linking to dbpedia.org[1] URIs or MusicBrainz[2] identifiers). We define `:mySimilarity` as the actual similarity statement. The `sim:element` property is used to refer to the tracks involved in this similarity and the `foaf:maker` property refers to the agent which asserted this similarity. Also note we can assign a numerical weight value to the similarity using the `sim:weight` property.

[1] http://dbpedia.org
[2] http://musicbrainz.org/

```
:track01 a mo:Track .
:track02 a mo:Track .

:me a foaf:Person .

:mySimilarity a
  sim:Similarity ;
  sim:element :track01 ;
  sim:element :track02 ;
  sim:weight "0.90" ;
  foaf:maker :me .
```

Listing 5.1: A simple example of a similarity statement using MuSim.

Now we have a method for asserting a similarity statement and reifying that statement to some extent. However, in the above example we only know *who* is making the similarity statement, we do not know *how* or *why*. But before we extend our modeling to better deal with these questions, let us further discuss reification.

Reification of statements

It is worth noting that our approach to association reification is closely related to the reification approach provided by the `rdf:Statement` concept where a triple with a subject, property, and object is modeled as a single entity. An `rdf:Statement` instance must have a `rdf:subject`, `rdf:property`, and `rdf:object` property. What would normally be a single triple is modeled as a set of four triples. This is illustrated in listing 5.2.

This overhead allows us to bind additional information to a statement for example attributing the statement to a particular individual or assigning a time stamp to the statement. However, this approach to reification brings about some problems as discussed by Miller [2000]. For one, although the two statements in listing 5.2 are syntactically unique, in the RDF model they are exactly equivalent. And in the RDF model all statements are considered members of the *set* of statements. By definition, sets cannot contain duplicates. In this sense the will to use RDF as its own meta-system begets an implementation paradox that would result in the loss of information - what happens to additional properties we bind to `:Statement` when it is necessarily reconciled with the simple triple syntax?

```
# simple triple
:Galactic mo:similar_to :TheMeters .

# reified triple
:Statement a rdf:Statement;
  rdf:subject :Galactic ;
  rdf:property mo:similar_to ;
  rdf:object :TheMeters .
```

Listing 5.2: A simple triple and a reified triple statement

The solution most widely adopted for this problem is related to named graphs where a given set of triples is grouped together and assigned some identifier. Then statements about that set of triples can be made using the identifier in additional triple statements. This means in many implementations a triple is actually stored as a quad, adding the name of the graph to which the triple belongs [Carroll et al., 2005].

The use of `rdf:Statement` has largely fallen out of favor although it is still part of the official W3C recommendation [Beckett, 2004a]. The Similarity Ontology enables a syntax that is reminiscent of `rdf:Statement` reification especially when talking about a *directed* association. An example of directed similarity modeling in MuSim is given in listing 5.3.

```
:DirectedAssoc a sim:Association ;
  sim:subject :TheMeters ;
  sim:object :FunkyMeters .
```

Listing 5.3: Basic directed similarity in MuSim.

This syntax resembles an `rdf:Statement` where the `rdf:property` is implicit. However, we must note that `sim:Association` is *not* a sub-class of `rdf:Statement` and does *not* inherit any of its characteristics. Therefore a `sim:Association` is not equivalent to some singular triple statement, avoiding the duplication problems inherent to `rdf:Statement`. Furthermore, `sim:subject` is *not* a sub-property of `rdf:subject` and `sim:object` is *not* a sub-property of `rdf:object`.

So why use a syntax reminiscent of an unpopular modeling paradigm that has fallen out of favor? Why not simply use the more fashionable named graphs approach?

Although reification of similarity statements is possible with named graphs, in our model we enable a different approach that, as we will see later, still employs named graphs but not in the context of association statement reification. Instead we apply the `sim:Association` concept for reification. As we will see the two approaches turn out to be nearly equivalent in terms of complexity and the MuSim approach tends to be more "implementation friendly". The named graph approach and the MuSim approach to modeling an undirected similarity are compared in listing 5.4.

```
# (a) named graphs similarity
:NGSim { :Galactic mo:similar_to :TheMeters .
         :TheMeters mo:similar_to :Galactic . }
:NGSim foaf:maker :me ;
  sim:distance "3".

# (b) MuSim similarity
:MuSimSim a sim:Similarity ;
  sim:element :Galactic ;
  sim:element :TheMeters ;
  foaf:maker :me ;
  sim:distance "3" .
```

Listing 5.4: Reification of similarity with (a) named graphs and (b) MuSim

In this example the named graphs approach results in four triples and one graph specification, while the MuSim approach results in five triples and no named graphs. If, for the purposes of complexity measurement, we consider a named graph specification approximately equivalent to a triple specification, the two approaches have roughly equivalent complexity. The named graphs approach results in a valid RDF dataset that can be queried by SPARQL as does the MuSim approach. However the addition of the graph specification means that listing 5.4 (a) is not valid Turtle syntax. The TriG[3] syntax extends Turtle with facilities for named graphs and listing 5.4 is valid TriG while listing 5.4 (b) is valid in Turtle and TriG.

The advantages of MuSim reification become more clear when we consider triple store implementations. Consider we have N entities and we want to specify $(N-1)^2/2$ undirected associations. Furthermore we want to bind a `sim:distance` value to each association. With the named graphs approach

[3] see http://www4.wiwiss.fu-berlin.de/bizer/TriG/Spec/

we would end up specifying $(N-1)^2/2$ graphs and with the MuSim approach we would specify as many `sim:Associations`. In the triple store implementation each named graph would require an explicit name that must be stored as some hash along with references to the triples in the graph and references to triples involving the graph name. The `sim:Associations` would simply require references to triples involving the given association. Furthermore, if we assume we will refer to associations not by some arbitrary name, but by the identifiers of the entities involved in the association statement, all `sim:Associations` can be unnamed blank nodes. By definition, named graphs cannot be blank nodes. This means in the triple store implementation our association statements can be stored with a simple and scalable integer index rather than some hashing method. Such an approach is employed by the popular 4Store[4] triple store software.

5.2.2 Association Networks

Note there is a strong analogy between our ontological framework for association and the complex network modeling approach we discussed in chapter 2 and applied in chapter 3. An instance of `sim:Association` is analogous to an *edge* in a network. The entities bound to the `sim:Association` instance are analogous to *nodes*. Nodes can be any RDF resource. The use of the `sim:element` predicate implies an undirected edge while the use of the `sim:subject` and `sim:object` predicates imply a directed edge. Associations that involve more than two entities are possible in our framework. In complex network parlance these are called *hyper edges*.

To affirm this analogy we create the `sim:Network` concept which defines a network by simply specifying a collection of `sim:Association` instances. The `sim:Association` statements in turn specify the RDF nodes involved in the network. In this way, we can use the graph-based structure of RDF to enable the specification and annotation of network structures - we are using a graph model to describe another graph at a higher level of abstraction. In addition to providing this wonderfully meta network description framework, the `sim:Network` concept provide a convenient container for grouping a set of `sim:Association` statements.

[4]see http://4store.org/

5.2.3 Association Methods

With the `sim:Association` concept and its sub-concept `sim:Similarity` we have developed our approach to reification of connections between entities. We now introduce the `sim:AssocationMethod` concept to identify the process used to derive association and similarity statements. It is here that we can tailor our framework to the domain of music. By describing the methods used to derive associations we make our very general approach more domain-specific. As we will see this approach allows us to encode a significant amount of detail, while maintaining the ability to write concise and efficient SPARQL queries against our data. We instantiate `sim:AssociationMethod`s that describe certain approaches to music similarity. This concept enables some interesting functionality when consuming the associations data - a consumer application can elect to include only similarity statements that are derived by a particular process. Consumption of MuSim data is discussed further in section 5.3.1. For now let us consider listing 5.5.

```
:timbreSimilarityStatement a sim:Similarity ;
  sim:element :track01 ;
  sim:element :track02 ;
  sim:weight "0.9" ;
  sim:method :timbreBasedSimilarity .

:timbreBasedSimilarity a sim:AssociationMethod ;
  foaf:maker :me ;
  sim:scope mo:Track ;
  dc:description """An algorithmic method for audio-based
                    similarity based on timbre""" .
```

Listing 5.5: An example of a similarity statement bound to an association method.

Here `:timbreBasedSimilarity` is the entity that describes our process for deriving similarity statements. Note that this entity is only described by three triples - its class type, a property for the creator (`foaf:maker`) and a brief textual description (`dc:description`). In a real-world application, a potentially large plurality of association statements would be bound to a single `sim:AssociationMethod` instance. Instead of binding various information about who is responsible for the association statements and how they were derived to each individual statement, we create an instance of

`sim:AssociationMethod` and bind the appropriate statements to this instance.

In a sense an instance of `sim:AssociationMethod` is analogous to defining a *class* of similarity statements. We extend this analogy by allowing the specification of the scope of a particular `sim:AssociationMethod`. For example, some methods will only apply to recorded audio signals and other methods will only apply to music artists. This can be specified by binding a `sim:scope` predicate to a particular `sim:AssociationMethod` instance as shown in listing 5.5. The use of the `sim:scope` predicate implies that a particular method yields undirected association statements. For directed methods, the `sim:domain` and `sim:range` predicates are applied. Note that while these properties resemble `rdfs:domain` and `rdfs:range` in name and function, they are not sub-properties of these properties. This avoids some of the reification issues described in section 5.2.1.

5.2.4 Provenance

We can use the `sim:AssociationMethod` concept introduced in section 5.2.3 to provide provenance for similarity statements.

Provenance, from the French word "provenir" meaning "to come from", descibes the origin and lineage of an entity. Provenance has become a very important topic for the web and it has become a topic of acute interest with respect to semantic web and linked data technologies. If automated software agents are to make use of linked data, the provenance of the data consumed becomes of utmost importance. A software agent that consumes the linked data equivalent of spam will likely have its utility undermined. We must be able to automatically follow the provenance of statements from various knowledge bases within a provenance ecosystem that is resilient to spam and forgery.

As of this writing there are a variety of purposed provenance solutions for the linked data web in the form of web ontologies. Examples of provenance ontologies and frameworks include the Open Provenance Model[5], the Provenir Ontology[6], the Provenance Vocabulary[7], and elements of the Proof Markup Language[8]. The W3C has formed an incubator group[9] to work towards recommendations for provenance on the semantic web. The group

[5]see http://openprovenance.org/
[6]see http://wiki.knoesis.org/index.php/Provenir_Ontology
[7]see http://trdf.sourceforge.net/provenance/ns.html
[8]see http://tw.rpi.edu/portal/Proof_Markup_Language
[9]see http://www.w3.org/blog/SW/2010/04/14/first_reports_of_the_w3c_

has purposed a set of provenance dimensions, more than 30 use cases and over 250 technical requirements. However, no candidate recommendation has been drafted as of this writing.

We can examine the dimensions specified by the W3C incubator group and the concepts that are common across the variety provenance ontologies already published. At the most abstract level we have three key concepts in provenance:

- the *Artifact* - some immutable piece of state that will most commonly refer to some digital representation of data
- the *Process* - some action or series of actions performed on or caused by artifacts and resulting in new artifacts
- the *Agent* - some contextual entity (i.e. a person or institution) acting as a catalyst enabling, facilitating, controlling or somehow affecting the execution of a process

The details of how these concepts inter-relate vary somewhat from implementation to implementation but these concepts are present in some form in all the aforementioned frameworks.

Given that the topic of provenance with respect to the linked data web is currently in a state of flux, we leave a full and rigorous integration of the Similarity Ontology with a provenance framework to future work. However, such integration can be considered rather trivial if we consider that the Artifact, Process, and Agent concepts will be a part of any useful provenance framework. The `sim:AssociationMethod` concept corresponds to a Process in a provenance framework and the `sim:Association` concept corresponds to an Artifact. We would simply treat the `sim:AssociationMethod` concept as a Process concept in the given provenance framework and treat the `sim:Association` concept as analogous to the Artifact concept in the given provenance framework.

In the interim, we provide a very basic method for dealing with provenance in the Similarity Ontology by employing some simple concepts from the FOAF Ontology[10]. We quite simply bind some `foaf:Agent` to a given `sim:AssociationMethod` using the the `foaf:maker` property. In this way we simply state *who* is behind a given similarity derivation process. This approach is practical but simplistic and provides very little recourse for avoiding spam or forgery of similarity statements. However, this simple approach

`provenance_incu`

[10]see http://xmlns.com/foaf/0.1/

could perhaps be sufficient in combination with the WOT RDF vocabulary[11] which allows agents to sign statements using Public Key Cryptography. Further explorations of provenance issues are beyond the scope of this thesis and left to future work.

5.2.5 Workflows

With the `sim:AssociationMethod` concept we have seen how we can make statements about the nature and provenance of a large plurality of association statements. We can potentially go even further and make these methods transparent by including information about the association derivation workflow. We will bind a description of a workflow to an instance of `sim:AssociationMethod` using the `sim:workflow` property. First we will discuss some previous work related to workflow modeling.

Workflow Modeling Frameworks

Workflow systems have attracted considerable attention in the research community [Taylor, 2007]. Scientific workflows that enable *in silico* experimentation are of paramount interest. The science of workflow management began with the automation of business processes and has evolved into one of the most important areas of e-science with an emphasis on the facilitation of process re-use, the leveraging of distributed resources, and the sharing of practical know-how. A recent informal search discovered over 75 unique workflow management systems on the web [De Roure et al., 2009]. We will only review a handful of the most relevant of these systems here.

Many of the most popular workflow management systems provide a rich graphical user interface allowing users to author workflows visually. Examples include the Kepler Project[12] [Altintas et al., 2004], Taverna [Oinn et al., 2004], and Trident [Barga et al., 2008]. These projects are of particular note as they excel at the creation of *workflow templates* and the translation of these templates to *workflow instances*. A workflow template describes the steps and order of the process without identifying particular end points of service or execution code while a workflow instance binds to concrete executions. The Kepler Project provides a java-based application that allows the user to specify a workflow involving remote data sources and computational resources and then execute the workflow. The Taverna framework

[11] `http://xmlns.com/wot/0.1/`
[12] see `http://kepler-project.org`

has a focus on bioinformatics and facilitates the use of open resources available on the web. These resources tend to be highly heterogeneous and there is no contract in place to ensure quality of service. However, the Taverna framework is able to accommodate this heterogeneity and provides over 3500 bioinformatics-oriented operations.

While the frameworks mentioned so far focus on constructing workflow templates and executing workflow instances, the myExperiment Virtual Research Environment focuses on sharing and re-using workflows [De Roure et al., 2009]. The myExperiment site leverages some semantic web technologies to enable a scientific workflow-centric social network where users publish, share, and re-use scientific workflows. A *multiworkflow* approach is adopted allowing users to specify workflows in any manner they deem appropriate. This approach anticipates a future where workflows from various frameworks can be reconciled providing maximum interoperability. However, at the time of writing, most myExperiment users seem to favor the Taverna workflow modeling approach and, to the best knowledge of the authors, very little inter-framework interoperability currently exists.

A variety of workflow frameworks with an emphasis on music analysis have been developed. The knowledge management system purposed by Pachet [2005] makes use of a data model very similar to that of RDF and allows for fields[13] to be computable. However, the system is not explicitly intended for workflow modeling and it is a centralized system. The music-to-knowledge (M2K) provides a music informatics prototyping and evaluation platform that allows for the graphical specification of workflows operating on multimedia data [Downie et al., 2005]. Operations are visualized as nodes in a graph structure with inbound and outbound links specifying data flows. Workflows can be executed within the M2K system, however the sharing of workflows externally is not possible. The MARSYAS framework provides robust facilities for distributed music processing. However the dataflow networks used in MARSYAS are closely coupled to implementation details and do not provide an abstract workflow model.

The music knowledge management framework first purposed by Abdallah et al. [2006] is centered around a "knowledge machine" that can execute various processing steps on demand and table the results for later re-use. This system later evolved into a concrete workflow representation framework based on an extension of N3 called N3-Tr [Raimond, 2009]. Recall that N3 (discussed in section 4.2.4) is an extension of the RDF model that allows for the existence of RDF graphs (a set of triple statements) as quoted of formulæ.

[13]Here a field corresponds to an object in an RDF triple statement.

In N3-Tr, special "built-in" predicates specify a workflow process. The inputs and outputs of processes are specified along side these built-in predicates and grouped together as graphs. Concepts from concurrent transaction logic [Bonner and Kifer, 1996] are added to N3 to allow for operations within a graph to occur concurrently using the predicate `ctr:cc`. The Henry resource described in section 4.6.1 is an implementation of an N3-Tr agent. We make use of the N3-Tr framework in the example workflows we present here.

Rules and Proofs

In some instances an association is based on a complex workflow for example in the context of audio-based similarity between audio signals. In other instances an association is based on something much simpler, for example two music artists might share an association based on the fact they are from the same hometown. In these simpler scenarios a full-blown workflow framework is probably not appropriate.

The extensions of the RDF model provided by N3 allow for the specification of inference rules using the predicate `log:implies`. We have described the N3 model in section 4.2.4 and additional information can be found in Berners-Lee et al. [2007]. We make use of N3 rules in chapter 6 for evaluating our framework against similarity models from cognitive psychology.

In addition to N3 rules, a vast variety of rule languages exist and many are intended to interoperate with RDF. The Rule Markup Language (RuleML) is a markup language developed to express rules in XML for deduction, rewriting, and inferential transformations [Boley et al., 2001]. The SWRL framework [Horrocks et al., 2004] is a W3C submission which combines OWL and RuleML to provide a rules language for the semantic web. The Proof Markup Language provides a sort-of middle ground allowing for a simplified workflow modeling in that it allows us to relate a conclusion with the corresponding premises and axioms used to derive the conclusion [da Silva et al., 2006]. More recently, the Rules Interchange Format (RIF) has become a W3C recommendation. The aim of RIF is to provide a means for exchanging rules between the multitude of rule languages and systems that already exist [Boley et al., 2010]. The rule languages we've mentioned here are all based on the XML syntax and other rule languages can be specified using some dialect of RIF. This means any of these rule languages could be applied to our associations framework using the `rdf:XMLLiteral` type.

Workflows and MuSim

We use the `sim:workflow` property to bind an association method to a workflow template. Similar to the open-ended approach adopted by myExperiment [De Roure et al., 2009], we do not place explicit restrictions on how the workflow should be specified. However, in this work we make use of N3 rules for simpler scenarios and N3-Tr for more complex workflows.

We can extend listing 5.5 by binding a workflow template to the association method. This is done in listing 5.6:

```
:timbreSimilarityStatement a sim:Similarity ;
  sim:element :track01 ;
  sim:element :track02 ;
  sim:weight "0.9" ;
  sim:method :timbreBasedSimilarity .

:timbreBasedSimilarity a sim:AssociationMethod ;
  foaf:maker :me ;
  dc:description """An algorithmic method for audio-based
                    similarity based on timbre""" ;
  sim:workflow :algorithm .

:algorithm = {
{ { ?signal1 mo:published_as ?track01 .
    ?signal1 sig:mfcc ?mfcc1 .
    ?mfcc1 sig:gaussian ?model1 }
      ctr:cc
  { ?signal2 mo:published_as ?track02 .
    ?signal2 sig:mfcc ?mfcc2 .
    ?mfcc2 sig:gaussian ?model2 } .
  (?model1 ?model2) sig:emd ?div .
  ?div math:lessThan "0.2" } =>
  { _:timbreSimilarityStatement
      a sim:Similarity ;
      sim:element ?track01 ;
      sim:element ?track02 }
}
```

Listing 5.6: An association method described by a workflow graph.

In the above example, when we follow the `sim:workflow` property we see an RDF graph `:algorithm` denoted by the `{` and `}` characters. This N3-Tr graph provides a disclosure of the algorithm used in the similarity derivation process. In this case, Mel-frequency cepstral coefficients (MFCCs) are extracted and Gaussian mixture models are created concurrently for the two signals, and an earth mover's distance is calculated between models. This is a method commonly used in audio-based signal analysis to derive some measure of timbre similarity [Logan and Salomon, 2001]. Depending on that distance, we output a similarity statement. If more details are needed about a particular computational step, e.g. if we want to gather more information about the MFCC extraction step, we can look-up the corresponding web identifier, in this case `sig:mfcc`.

Here, the N3-Tr formulæ describe the workflow supporting the similarity statement. Given we are already using a superset of N3, we could fore-go the use of the `sim:AssociationMethod` concept and use the `log:supports` predicate in the N3 framework [Berners-Lee et al., 2007]. However, as we will discuss in section 5.3.1, binding similarity workflows to the `sim:AssociationMethod` concept allows us to make simple, useful queries (i.e."show me all similarity derivation methods available in the system"). Furthermore processing workflows as part of a query can be complicated due to the lack of interoperability between workflow frameworks as we discussed in section 5.2.5.

In listing 5.6 we make our association method transparent by specifying an appropriate N3-Tr workflow. Alternatively we can provide a minimum amount of information when dealing with a "black box" similarity derivation processes. Various levels of transparency are illustrated in figure 5.3.

As indicated in 5.3, our framework also supports the grounding of similarity statements directly through the property `sim:grounding`. This property associates a similarity statement with the instantiated workflow. When using `sim:grounding` we link our association statements directly to a specific workflow instance with references to the calculated values at each step. Such a grounding allows us to filter similarity statements on the basis of specific features of their computation. In the grounded case, we detail explicitly the results obtained at each step of the workflow. An example of grounded similarity statement is provided in listing 5.7.

Again, we are using the audio-based timbre similarity workflow but in this case we know the numerical values of the intermediate calculations. For example, we can see that the result of the earth mover's distance calculation is 0.1.

```
:timbreSimilarityStatement sim:grounding
{
 { _:signal1 mo:published_as :track01 .
   _:signal1 sig:mfcc _:mfcc1 .
   _:mfcc1 sig:gaussian _:model1 }
     ctr:cc
 { _:signal2 mo:published_as :track02 .
   _:signal2 sig:mfcc _:mfcc2 .
   _:mfcc2 sig:gaussian _:model2 } .
 (_:model1 _:model2) sig:emd 0.1 .
 0.1 math:lessThan 0.2 }
}
```

Listing 5.7: An example of a similarity statement with some provenance and transparency.

5.3 A Similarity Commons

The data model provided by the Similarity Ontology allows for lots of flexibility in specifying similarity statements. This flexibility is balanced by the inclusion of provenance tracking and transparency. By following the `sim:method` property in a similarity statement we can find information about *who* made the statement and *why*. When consuming similarity data, we select statements by deciding which agents and algorithms to trust. While it is entirely possible to make a similarity statement within this framework completely anonymously, such statements are likely to be ignored by data consumers. Instead the statements from trusted agents or transparent algorithmic processes are likely to be selected by data consumers. In a music recommendation application, this allows for more transparent recommendations - providing the end user with the source or process used to make the recommendation. Intuition as well as recommender system research suggest users are more likely to trust transparent recommendation processes [Celma, 2008].

Beyond the specification of the Similarity Ontology, we envision a broader music similarity commons where autonomous, semi-autonomous, and human agents operate in tandem, making similarity statements about music tracks and artists while providing provenance and justification for these statements. A simple diagram illustrating how this commons might be structured is provided in 5.4.

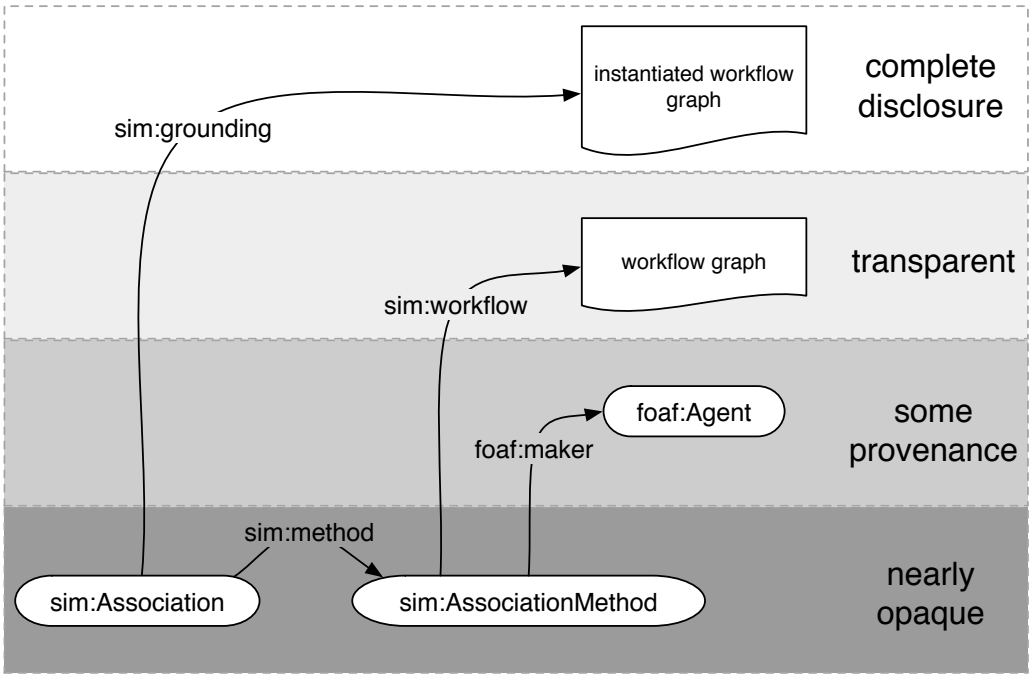


Figure 5.3: Using the Similarity Ontology. As additional properties are bound to our association and association method statements, we achieve greater transparency.

An enabled client music application publishes the end user's listening habits to the web. Similarity agents operate on the web and publish their own music similarity statements - perhaps consuming the listening habits of end users as well as other data. These statements refer to specific URIs for each track and artist. Similarly, the client music application links the content in the user's personal collection to URIs using methods such as those detailed in [Raimond et al., 2008]. This avoids ambiguity - we can be sure that the similarity statements are referring to the specific resource in which we are interested. The similarity statements made by various agents are aggregated into one or more data stores for querying. The client music application, perhaps responding to a user request, can query the data store for similarity statements from trusted agents involving the target resource (e.g. a track or artist). The query returns similarity information that can be used for content recommendations or playlist generation.

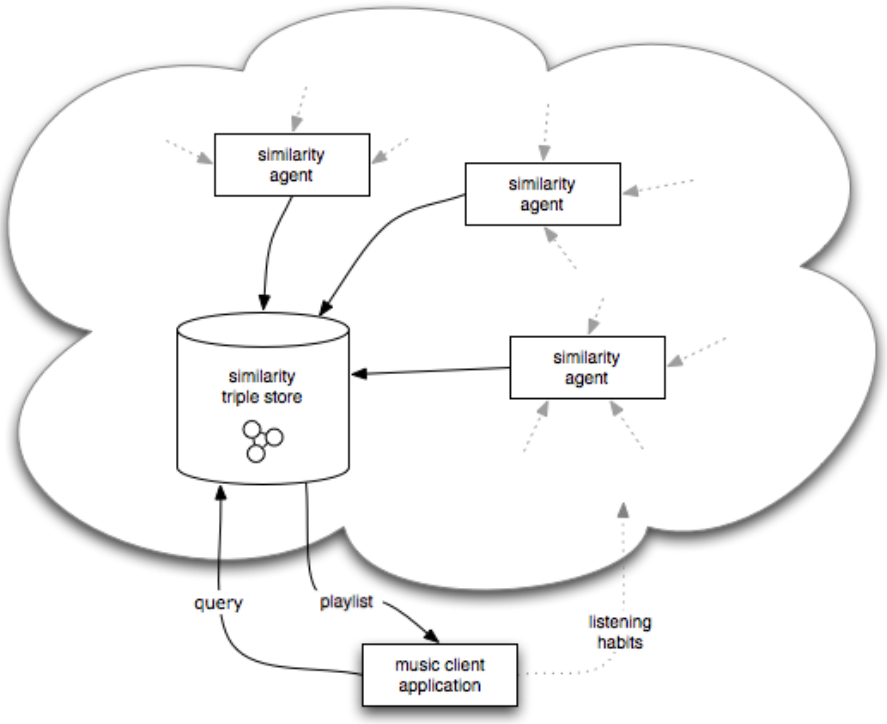


Figure 5.4: The music similarity commons. Similarity agents operate on structured data to create similarity statements. Such statements are aggregated in a data store and queried by a client music application to provide recommendations, playlists, and other functionality.

5.3.1 Similarity Queries

Queries in this similarity commons would be made using the SPARQL query language which we discussed in section 4.5. The SPARQL specification is a W3C recommendation and the preferred method for querying RDF graphs. As mentioned before, the design of the Similarity Ontology allows for the construction of simple queries to retrieve similarity information. The query in listing 5.8 retrieves artists similar to a target artist as stated by a specific trusted method.

Notice we only have to include a triple pattern for our target resource, a triple pattern for our trusted agent, and a triple pattern to select the similar artists. Of course this is a very simple example and in real-world applications we include additional optional patterns and conjunctions for a more expressive query.

In an initialization step, an application could query available data sources

```
PREFIX sim: <http://purl.org/ontology/similarity/>

SELECT ?artists WHERE {
  ?statement sim:method <http://trusted.method/uri> ;
    sim:element <http://target.artist/uri> ;
    sim:element ?artists .
}
```

Listing 5.8: A SPARQL query for similarity statements specifying a trusted derivation method.

to determine exactly what association methods and asserting agents are available. The application would use a query similar to that in listing 5.9.

```
PREFIX sim: <http://purl.org/ontology/similarity/>

SELECT DISTINCT ?method WHERE{
  ?method a sim:AssociationMethod .
}
```

Listing 5.9: A SPARQL query to retrieve a list of available association methods.

The application could then filter through the results and, perhaps with some input from the end-user, decide which similarity agents to trust.

5.3.2 Similarity and Recommendation

While we hold that similarity is the basis of recommendation, we also acknowledge that similarity and recommendation are not identical. By no means does the similarity commons proposed here solve the problems of recommender systems - rather it provides a new distributed cross-domain platform on which future recommender systems might be built.

While an item-to-item recommendation system fits quite naturally into this similarity commons, we can also imagine a collaborative filtering-style user-item recommendation system. Each user in the system is treated as an `sim:AssocationMethod` instance. Each user's method makes a set of statements asserting that the tracks found in that user's personal collection are similar to each other. Then an additional `sim:AssocationMethod` instance is

used to match users with each other based on the contents of their respective music libraries. Finally, for a given user, the recommendations for that user are an aggregation of the similarity statements derived from the association methods bound to the most similar users.

Also note that the similarity commons fosters hybrid recommendation approaches. Because the similarity statements are made using common semantics and syntax, we can easily combine and compare these statements to derive recommendations or new similarity statements.

5.4 Summary

We have presented an ontological framework for describing similarity statements on the web of data. This ontology is extremely flexible and capable of expressing a similarity between any set of resources. This expressiveness is balanced by transparency and provenance, allowing the data consumer to decide what similarity statements to trust. We have shown hows this framework could exist as the foundation for a broader music similarity commons where autonomous, semi-autonomous, and human agents publish a wealth of similarity statements which are combined, consumed, and re-used based on provenance, trust, and application appropriateness.

We have suggested how similarity algorithms can be made transparent. We have adopted the N3-Tr syntax for describing similarity derivation workflows. In future work we plan to extend this syntax and the supporting ontologies to better enable the publication of similarity derivation workflows. Furthermore we hope to develop a series of recommendations for best practice when publishing such workflows to maximize their usefulness and queryability.

While our Similarity Ontology was designed with music similarity in mind, it is by no means limited to the domain of music. We leave it to future work to explore how this framework might be applied in different domains and across domains. The similarity framework described here is made specific to music in the instantiation of `sim:AssociationMethod`s and other supporting concepts that further define a similarity statement. The same approach could be applied to other domains such as literature, film, or entertainment.

An evaluation of the Similarity Ontology and our associations framework is provided in chapter 6.

Chapter 6

Evaluation

> *Whenever there is a simple error that most laymen fall for, there is always a slightly more sophisticated version of the same problem that experts fall for*
>
> –Amos Tversky, Israeli cognitive psychologies (1937-1996)

In chapter 5 we developed the Similarity Ontology and presented our vision for a web-scale similarity commons. Here we provide some evaluation of this ontology and the similarity framework. The design of the Similarity Ontology is minimalistic and it provides only a very few broad concepts. Of course this can be viewed as both a strength and weakness of the ontology. When discussing the similarity framework we include the formalisms, methodologies, and some of the additional vocabularies that exist in the linked data world. In this sense our evaluation is still an ontology evaluation but now includes not only the Similarity Ontology but additional ontologies as well.

We briefly review a variety of ontology evaluation methods in section 6.1 and then adopt a task-based approach to show how the Similarity Ontology, along with the expressiveness of N3, is flexible enough to accommodate the most popular psychological models for similarity perception in section 6.2. In section 6.4 we show how the music artist networks described in chapter 3 could be modeled in a uniform manner in our similarity ecosystem.

While we provide a direct task-based evaluation of our similarity framework in this chapter, we provide an indirect evaluation of our framework in section 7.5.3 via a system usability scale survey for an end-user application based on the Similarity Ontology.

6.1 Ontology Evaluation Techniques

Ontology evaluation is an important topic with respect to the semantic web - we must have some notion of which ontologies are most appropriate to use and re-use for particular applications. To this end a variety of domain and task-independent evaluation methodologies have been developed [Vrandečić, 2009]. These methods focus on the structure of ontologies and develop metrics based on network analysis tools similar to those we applied to music artist networks in chapter 3. For example the AKTiveRank system develops a set of metrics based on the number of property edges incident to a concept (degree) and the centrality of concepts in the ontology structure (betweeness) [Alani and Brewster, 2006]. However, these methods necessarily ignore the quality of the domain modeling. Given our ontology is very small and minimalistic such evaluation methodologies are not particularly apt.

Alternatively domain-specific ontology evaluation methods focus on how well a given ontology models the target domain. A variety of domain-specific ontology evaluation methods are surveyed in [Obrst et al., 2007]. For example we might use a set of criteria to have humans assess an ontology by hand as in [Smith et al., 2005]. However, this is a time consuming process that involves human subjects and the minimalistic nature of the Similarity Ontology would make formulating appropriate questionnaires difficult.

Instead we focus on a task-driven evaluation of MuSim as advocated by [Brewster et al., 2004]. We use two major tasks for this evaluation. First in section 6.2 we attempt to compare our modeling to the reality of similarity perception as described in the psychology literature. We describe four different models of similarity cognition pulled from the cognitive psychology literature and show how the Similarity Ontology can accommodate these models. Second we revisit the variety of music artist networks found on the web that we discussed in chapter 3 and show how MuSim can be used to model these artist networks.

6.2 MuSim and Psychological Models of Similarity

Similarity has long been an important topic in experimental psychology. Human judgements of similarity are a fundamental part of cognition - our sense of similarity allows us to order things in to kinds and in this respect is similarity judgements enable learning, knowledge, and thought. This philosophical

perspective on the importance of similarity is better articulated in a variety of important works [Shepard, 1982; Tversky and Gati, 1982; Quine, 1969; Tenenbaum, 1996].

A multitude of experimental studies have examined the topic of human similarity judgements [Heit and Rubinstein, 1994]. These experimental studies have given rise to a variety of models that explain human similarity judgements. We will examine the four most pervasive models of similarity as selected by [Goldstone and Son, 2005]: geometric models, featural models, alignment-based models, and transformational models.

As a means of evaluating the flexibility of our similarity framework, we attempt to impose the constraints of each psychological model with the tools available in our framework. It should be noted that the modeling examples provided here are probably not in-line with what a pratical application might employ. Instead, these are intended to demonstrate the flexibility of the system.

6.2.1 Geometric Models

Geometric models of similarity have been perhaps the most influential approaches to analyzing similarity. In geometric models relations between entities are modeled with some distance function that satisfies the triangle inequality. That is where A, B, and C are some entities and $d(A, B)$ is dissimilarity between A and B we have

$$d(A, B) + d(B, C) \geq d(A, C) \tag{6.1}$$

The statistical techniques of multidimensional scaling are often applied to a set of empirical pairwise dissimilarity judgements to attempt to determine the dimensions and dimensional values that subjects use to make similarity judgements [Richardson, 1938; Torgerson, 1965; Shepard, 1982].

Note that a set of entities and an inter-entity distance function that satisfies the triangle inequality constitutes a *metric space*. There is a wide body of research and some very important applications in computer science that involve searching over metric spaces [Chávez et al., 2001].

We can partially model a metric space scenario quite easily using MuSim RDF. Let us suppose we have a set of three entities we'll call `:A`, `:B`, and `:C`.

However, this modeling does not impose the triangle inequality requirement. We can use the added expressiveness of N3 to impose the triangle inequality as in listing 6.2.

```
:Similarity0 a sim:Similarity;
  sim:element :A ;
  sim:element :B ;
  sim:distance "5.0"^^xsd:float .

:Similarity1 a sim:Similarity;
  sim:element :B ;
  sim:element :C ;
  sim:distance "4.0"^^xsd:float .

:Similarity2 a sim:Similarity;
  sim:element :A ;
  sim:element :C ;
  sim:distance "3.0"^^xsd:float .
```

Listing 6.1: Geometric similarity in MuSim is modeled simply by including an appropriate sim:distance value for each similarity statement. However without the expressiveness of N3 we cannot impose the triangle inequality restriction

Note that in our approach to modeling metric spaces with MuSim we are modeling distances between entities rather than modeling positions in the space. A special case of the metric space is the *vector space* where each entity can be modeled as a set of k real-valued coordinates in the space. A vector space approach is favored in many similarity search applications because the geometric properties of the space can be exploited. Only coordinate positions are stored and a series of inexpensive distance functions are calculated at query time [Chávez et al., 2001]. The Similarity Ontology does not provide a mechanism for modeling coordinates in a vector space. However, as we will see in section 7.4 it is quite possible to use MuSim RDF as an output layer for a vector space similarity database.

In summary, we can say that our similarity framework is capable of modeling geometric similarities but fails for the special case of vector space similarities.

```
{
  ?similarity0 sim:distance ?dAB .
  ?similarity1 sim:distance ?dBC .
  ?similarity2 sim:distance ?dAC .

  (?dAB, ?dBC) math:sum ?dAB_dBC .
  { {?dAB_dBC math:greaterThan ?dAC .}
      log:or
    {?dAB_dBC math:equalTo ?dAC . } } a log:Truth .
}
```

Listing 6.2: Using an N3 rule to impose the triangle inequality requirement.

6.2.2 Featural Models

Although geometric models of similarity provide an intuitive approach that enables some interesting and useful applications, there is a rather large body of empirical evidence that suggests human perception of similarity is actually incompatible with the geometric model assumptions. In addition to the triangle inequality restriction, the geometric model assumes minimality ($d(A, B) \geq d(A, A) = 0$) and symmetry ($d(A, B) = d(B, A)$).

Systematic violations of these assumptions have been found in experiments on human perception. For example, minimality requires that all objects are equally similar to themselves. But it has been shown that printed letters violate this assumption in terms of confusion rates [Nickerson, 1972]. It has also been shown that a non prominent item is perceived as more similar to a prominent item than vice versa [Tversky, 1977]. This violates the symmetry assumption of the geometric model. Finally, it has been shown that pair-wise similarity judgments simply do not conform to the triangle inequality assumption [Tversky and Gati, 1982].

These problems with the geometric model gave rise to a set of featural models for similarity. Similarity is characterized in terms of a feature-matching process based on weighting common and distinctive features [Tversky, 1977]. If we have entities A and B the similarity between these entities $s(A, B)$ is given by:

$$s(A, B) = \theta f(A \cap B) - a f(A - B) - b f(B - A) \tag{6.2}$$

where $(A \cap B)$ represents the features A and B have in common, $(A -$

B) represents the features A has but B does not, and $(B - A)$ represents the features that B has but A does not. Similarity is modeled as a linear combination of the measure of common and distinctive features where θ, a, and b are weight values that are context dependent. This specific formulation of the featural model is referred to as the linear contrast model. Alternative featural models formulate similarity as a ratio function rather than a linear contrast of features [Eisler and Ekman, 1959], however, here we will focus on the linear contrast model.

Let us now assume we have two entities identified by `:A` and `:B` and we want to make some assertions about their similarity that conforms with the linear contrast model. We can model this using the similarity ontology as shown in listing 6.3:

```
:Similarity a sim:Similarity;
  sim:subject :A ;
  sim:object :B ;
  sim:weight :SimilarityWeight ;
  sim:grounding :FeatureContrasts .

:FeatureContrasts = {
{
  :A ?pCommon ?oCommon .
  :B ?pCommon ?oCommon .

  :A ?pA ?oA .
  ?pA owl:differentFrom ?pCommon .
  ?oA owl:differentFrom ?oCommon .

  :B ?pB ?pB .
  ?pB owl:differentFrom ?pCommon .
  ?oB owl:differentFrom ?oCommon .

  :ABcommon a log:List;
    log:includes ?pCommon, ?oCommon .

  :AminusB a log:List;
    log:includes ?pA, ?oA .

  :BminusA a log:List;
    log:includes ?pB, ?oB .
```

```
  :ABcommon :f _:CommonFeaturesScore .
  :AminusB :f _:AFeatureScore .
  :BminusA :f _:BFeatureScore .

  (_:CommonFeaturesScore, _:AFeatureScore) math:difference _:diff .
  (_:diff, _:BFeatureScore) math:difference :SimilarityWeight .
} => :Similarity a sim:Similarity;
  sim:subject :A ;
  sim:object :B ;
  sim:weight :SimilarityWeight . }

:f a :builtInPredicate ;
   rdfs:comment """a function that returns a similarity score based on a
                   the size of a list of predicates and objects""" ;
   rdfs:range xsd:float .
```

Listing 6.3: Modeling featural similarity with the Similarity Ontology

The actual similarity statement itself is rather simple - a directed statement is made using the `sim:subject` and `sim:object` properties, some weight value is included, and an N3 graph is provided grounding our similarity statement. However, the grounding graph is somewhat involved. This graph is simply an implementation of the equation (6.2) in the N3 syntax. For simplicity we have assumed the weight values $\theta = a = b = 1$ and omitted them from the N3 graph - including weights would simply require a series of additional triples involving the `math:product` property. The property function `:f` is analogous to the f function in equation (6.2) and here is simply specified as a count of predicate-object pairs.

In summary, with the support of some basic N3-enabled algebra, our similarity framework is capable of accommodating featural models for similarity cognition.

6.2.3 Alignment-based Models

Both geometric and featural models neglect the structure of the features of the entities in question. Geometric models simply focus on inter-item similarity judgments and treat entities as monotonic. Featural models treat entities as a "bag of features" and neglect how these features are structured. More recent empirical data suggests that the structure of features within entities plays an important role in inter-item similarity [Holyoak and Koh,

1987; Goldstone, 1996; Markman and Gentner, 1993]. This has given rise to alignment-based models for similarity where features must not only match but also correspond, or align somehow to contribute to similarity.

The alignment-based view of similarity is perhaps the most compatible with the Similarity Ontology modeling. With MuSim we are quite capable of specifying a multitude of distinct similarities between a pair of items. Again, consider two entities identified by `:A` and `:B` respectively. We can satisfy the alignment-based model of similarity by simply modeling each aligned feature as an individual similarity statement as shown in listing 6.4.

```
:Similarity0 a sim:Similarity;
  sim:element :A ;
  sim:element :B ;
  sim:grounding :Sim0Grounding .

:Similarity1 a sim:Similarity;
  sim:element :A ;
  sim:element :B ;
  sim:grounding :Sim1Grounding .

:Sim0Grounding = {
  { :A :feature0 ?x .
    :B :feature0 ?x . } =>
  :Similarity0 a sim:Similarity;
     sim:element :A ;
     sim:element :B .
}

:Sim1Grounding = {
  { :A :feature1 ?y .
    :B :feature1 ?y . } =>
  :Similarity1 a sim:Similarity;
     sim:element :A ;
     sim:element :B .
}
```

Listing 6.4: Modeling alignment-based similarity with the Similarity Ontology and N3.

In listing 6.4 we see two distinct similarity statements identified by

:Similarity0 and :Similarity1 respectively. Both statements involve the same elements :A and :B and both statements are bound to a grounding graph. In the :Sim0Grounding we see that :A and :B share a common triple that involves the hypothetical property :feature0 and this implies a similarity. The :Sim1Grounding graph follows an almost identical pattern but applies to a distinct hypothetical property :feature1. Modeling these atomic similarities individually allows us to accommodate the alignment-based similarity model.

6.2.4 Tranformational Models

A final model for similarity judgments is the transformational model where similarity is modeled as the cost required to transform one stimuli into another. While this approach has some support from experimental psychology [Garner, 1974; Wiener-Ehrlrich et al., 1980; Imai, 1977], it is often more closely associated with artificial intelligence research [Ullman, 1996].

We can create a simple transformation cost function in an N3 graph. We simply model the transformation cost as the number of triples that are distinct for the entities in question. If we have items identified by :A and :B, how many triples must we add or subtract to make :B identical to :A. Such a formulation is shown in listing 6.5.

```
:Similarity a sim:Similarity ;
  sim:element :A ;
  sim:element :B ;
  sim:distance :transformation_cost ;
  sim:grounding :TransCalc .

:TransCalc = {
 {
  :A ?pA ?oA .
  :B ?pB ?oB .

  (?pA, ?oA) owl:differentFrom (?pB, ?oB) .

  _:triplesA a log:List ;
     log:includes (?pA, ?oA) ;
     list:length _:lenA .
```

```
   _:triplesB a log:List ;
      log:includes (?pB, ?oB) .
      list:length _:lenB .

   (_:lenA, _:lenB) math:sum :transformation_cost .
  } =>
  :Similarity a sim:Similarity ;
    sim:element :A ;
    sim:element :B ;
    sim:distance :transformation_cost .
}
```

Listing 6.5: Modeling transformation-based similarity with the Similarity Ontology and N3

In summary, by modeling transformation cost as the difference between the set of triples that apply to the entities in question, our similarity framework can accommodate the transformational model for similarity cognition.

6.3 Similarity Scenarios

We can also imagine some music similarity scenarios that might be expressed using our similarity framework. Many of these scenarios are drawn from the use-cases developed as a roadmap for the ontological modeling. Although they are included in this chapter, their only value with respect to evaluation is to show that our framework satisfies the use cases to which it was designed.

6.3.1 Multifaceted Audio-based Similarities

As first mentioned in section 3.2, a variety of techniques for algorithmically determining the musical similarities between audio signals have been developed [Logan and Salomon, 2001; Tzanetakis and Cook, 2002a; Berenzweig et al., 2004; Cano et al., 2005b; Jacobson, 2006; Pampalk, 2006]. In addition to methods for overall musical similarity, a variety of methods have been developed for extracting high-level musical features from audio signals, characterizing a music signal by its timbre [Logan, 2000], harmony [Bello, 2003], rhythm [Gouyon and Dixon, 2005; Davies, 2007], or structure [Levy et al., 2006].

In our similarity framework, we can combine similarity statements from a plurality of algorithms or even chain high-level musical features with a set of rules to derive similarity statements. In section 7.3 we develop a knowledge base that combines audio-based rhythmic similarity statements [Foote et al., 2002] along with audio-based timbre similarity statements [Levy and Sandler, 2006], and algorithmically derived tonality [Noland and Sandler, 2007] estimates to navigate a collection of public domain classical recordings.

6.3.2 Contextual Similarity

Because music is a complex construct deeply ingrained in culture and society, we might want to make music similarity statements that relate to the context of musical works rather than the content of the musical works themselves. Similar scenarios exist in other domains as well, but for now let us consider an example from popular rap music. In the mid to late 1980s a series of songs were released disputing the place of origin of the musical genre hip hop launching a multi-faceted feud that became colloquially referred to as *The Bridge Wars*[1]. By simply creating an association method that asserts similarities between artists and tracks related to this feud we can accommodate this scenario.

For example, Marly Marl and MC Shan released a track entitled "The Bridge" alleging hip hop started in Queens Bridge. South Bronx resident KRS One disagreed and responded by releasing a track entitled "South Bronx". We could encode this situation as shown in listing 6.6.

```
:rapFeudTracks
  a sim:Association ;
  sim:element [dc:title "The Bridge"; a mo:Track] ;
  sim:element [dc:title "South Bronx"; a mo:Track] ;
  sim:method :rapFeudSimilarity .

:rapFeudSimilarity
  a sim:AssociationMethod ;
  foaf:maker [a foaf:Agent] ;
  sim:description :algorithm ;

:algorithm = {
  ?track01 rap:response ?track02 }
```

[1] http://en.wikipedia.org/wiki/The_Bridge_Wars

Listing 6.6: The Bridge Wars modeled using MuSim and N3.

where `rap:response` represents the common practice of releasing a track as a response to another track.

6.3.3 Personal Associations

The emotional affect of music can be highly personal. A set of associations between music artists or tracks might be unique for one particular individual. Consider the following statement, "When a first year student at college, I dated a girl who listened to Bob Marley and David Bowie" - while this association between David Bowie and Bob Marley might hold weight for the narrator, it is likely that few other individuals would share this association. However, the narrator, for any number of reasons, may wish to express this association anyway. This is entirely possible in our ontological framework. The narrator can simply create an `sim:AssociationMethod` that asserts similarity statements based on the musical taste of his ex-girlfriend.

6.4 Artist Associations in the Similarity Framework

In chapter 3 we reviewed seven distinct resources on the web that could be used to construct music artist networks. In these networks a music artist is modeled as a node. Edges are formed between music artists based on the presence or absence of some relationship (e.g. collaboration, calculated similarity, influence, etc.). The web resources and the associated relationships between artists include:

- Classical Music Navigator - influence
- All Music Guide - editorial similarity, influence, collaboration
- Discogs - appeared on same release
- MySpace - top friend, audio-based similarity
- SoundCloud - following
- Echonest - web-mined similarity

- Last.fm - collaborative-filtering derived similarity

Again, we have seven distinct sources of information but because some websites provide multiple types of inter-artist relationships we have ten distinct relationship types. With the Similarity Ontology we model each relationship type as a `sim:AssociationMethod`. We use the `foaf:Agent` concept to model each source website and bind that to the appropriate `sim:AssociationMethod`. Then for each artist graph, each edge becomes a `sim:Association` statement (we use the broader `sim:Association` rather than `sim:Similarity` to accommodate the influence and collaboration relationships. Each association edge is then bound to the appropriate artist nodes (modeled as `mo:MusicArtist` instances) using the `sim:element` property for undirected edges or the `sim:subject` and `sim:object` properties if the edges are directed. An example of this modeling is provided in appendix B.

6.5 Summary

We have provided a task-based evaluation of the Similarity Ontology and the proposed similarity framework. It is shown that, when coupled with the expressiveness of N3, the Similarity Ontology can accommodate a plurality of the purposed models human similarity perception. The notable exception being the vector space approach - where items are modeled as points in space rather than modeling inter-item dissimilarities. The modeling of the various music artist networks found on the web using the Similarity Ontology provides another important task-based evaluation.

Additional evaluation of our framework is provided via the evaluation of a web application based on the Similarity Ontology. This user survey evaluation is described in section 7.5.3 and a variety of applications based on the work in this thesis are described in chapter 7.

Chapter 7

Applications

The wise musicians are those who play what they can master.

–Duke Ellington, American composer and musician (1899-1974)

We now turn our attention to describing several applications that have been enabled by the artist network analysis described in chapter 3, the semantic web technologies described in chapter 4, and the similarity framework described in chapters 5 and 6.

In section 7.1 we describe the Classical Music Universe application for visualizing and exploring influences between classical composers. In section 7.2 we develop the k-pie graph visualization algorithm and apply to the creation of an interface for exploring MySpace artists. In section 7.3 we describe a data set of public domain classical music recordings where audio-based similarities and composer influence have been modeled in MuSim. In section 7.4 we describe how audioDB (not developed as part of this thesis) utilizes the MuSim ontology. Section 7.5 describes the CatfishSmooth web application which aggregates linked data and re-publishes MuSim statements. In section 7.6 we further describe our extensions to the DBTune project. Finally in section 7.7 we describe the on-going LinkedBrainz project and describe how MuSim is applied to modeling Advanced Relationships from MusicBrainz.

7.1 The Classical Music Universe

The *Classical Music Universe*[1] is a tool for visualizing relationships and influence between classical composers. An RDF translation of composer relationships are derived from Charles H. Smith's Classical Music Navigator are used to construct an influence network - essentially identical to the network analyzed in section 3.4. The Classical Music Universe application leverages linked data from DBpedia.org and DBtune.org to provide additional biographical information about each composer.

7.1.1 Motivation

The motivation for the Classical Music Universe application was mostly related to exploration. Through its development the application was intended to explore several factors including:

- the challenges and advantages of building an application based on semantic web technologies;
- the utility of a graph-based user interface for music artist network navigation;
- the performance and utility of various graph layout techniques;

7.1.2 Implementation

The user interface for the Classical Music Universe is shown in figure 7.1. The main window of the interface contains a visualization of the composer influence network. Each circle represents a classical composer node. The size of each circle is logarithmically proportional to the in-degree k_{in} of the node in the composer influence network. The user can navigate the network by zooming and scrolling with the mouse and selecting focus on a composer with a mouse-click. Additional information about the highlighted composer is presented on the right hand column of the interface. This information is retrieved by SPARQL queries to DBTune and DBpedia - the application requires no database of its own relying instead on RDF graphs stored elsewhere. Only the coordinates for the network visualization and URIs for each node are stored locally by the client application.

[1] available at http://isophonics.net/cmu/ - note at the time of this writing this application is only supported in the Chrome browser

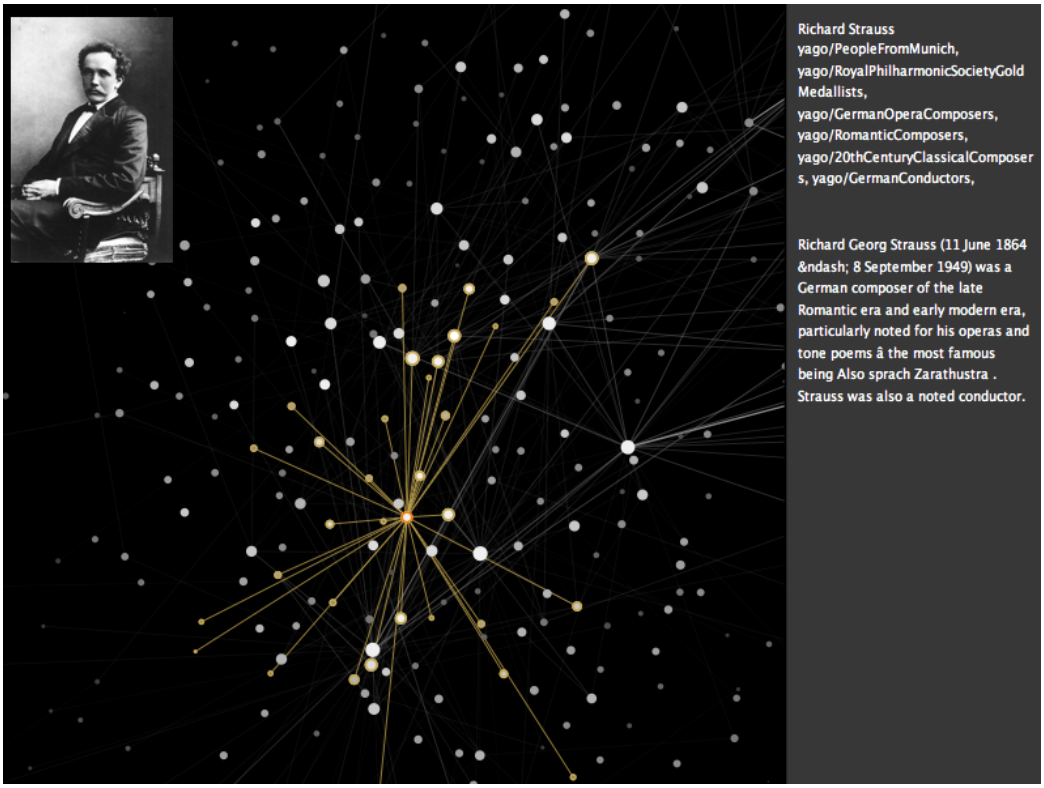


Figure 7.1: The Classical Music Universe composer influence visualization interface where each node represents a composer and each edge between nodes represents an influence relationship.

The first version of the application ran as Java stand-alone application using the Processing[2] visualization libraries. The final version is based on Javascript and the ProcessingJS[3] framework - running inside the web browser and making asynchronous calls to a Python web server that aggregates linked data.

In its original incarnation, the application was primarily a means of exploring the utility of different graph layout algorithms with respect to the visualization of small music artist networks (recall the CMN network only contains 426 nodes).

The Classical Music Universe implements three graph layout options. The k-cores decomposition layout [Alvarez-Hamelin et al., 2006], the Fruchterman-Reingold force-directed layout [Fruchterman and Reingold,

[2]see http://processing.org/
[3]see http://processingjs.org

1991], and the Kamada-Kawai layout [Kamada and Kawai, 1989]. The k-cores decomposition layout algorithm is based on dividing nodes into shells according to their k-coreness and positioning nodes within a shell in a co-centric circle. The k-core approach to graph decomposition is described in detail in section 7.2.1. The Fruchterman-Reingold and Kamada-Kawai layout algorithms are both examples of force-directed graph layout algorithms where physical models of forces between edges and nodes are applied to "push" elements of the graph into a state of equilibrium. In both algorithms, nodes are represented as steel rings and edges are springs between them. The attractive force is analogous to the spring force and the repulsive force is analogous to the electrical force. In the Fruchterman-Reingold algorithm, the sum of the force vectors determines the direction a node should move and a step width constant is set to determine how far a node should move in each iteration. A "global temperature" control is used to set the step width and ensure the algorithm terminates. The Kamada-Kawai algorithm achieves energy minimization by calculating the derivative of the force equations. When the derivative of the force equations are zero, minimum energy has been achieved. Of course the force equations are not independent so only the node with the maximum gradient value is moved in a single iteration. The process is repeated until the total energy is minimized.

7.2 K-Pie Graph Visualization

The k-pie algorithm is a rather intuitive extension of the network visualization algorithm k-core decomposition developed by Alvarez-Hamelin et. al in [Alvarez-Hamelin et al., 2006]. The k-core decomposition algorithm begins with a k-core analysis on a given graph structure and places vertices in a 2 dimensional space using a pair of polar coordinates - a radius related to the shellness of a given vertex and an angle related to the cluster of that vertex after k-cores analysis. We will review these concepts in more detail in section 7.2.1.

7.2.1 The K-Pie Algorithm

To describe the k-pie algorithm first we will provide some definitions and concepts associated with the k-core decomposition algorithm.

Definitions

Let us consider a graph $G = (V, E)$ where $|V| = n$ vertices and $|E| = e$ edges. As described in Batagelj and Zaversnik [2002], a *k-core* is defined as follows:

- A subgraph $H = (C, E|C)$ induced by the set $C \subseteq V$ is a k-core or a core of order k iff $\forall v \in C : degree_H(v) \leq k$, and H is the maximum subgraph with this property.
- A vertex i has a *shellness* c if it belongs to the c-core but not to the $(c+1)$-core. We denote the shellness of vertex i by c_i.
- A *shell* C_c is composed of all vertices whose shellness is c. The maximum value c such that C_c is not empty is denoted c_{max}. The k-core is then the union of all C_c with $c \geq k$.

It is important to note the distinction between a k-core and a k-shell - a k-shell implies a certain range of vertex degrees while a k-core only implies a lower limit to vertex degree. We are more interested in k-shells for our visualization algorithm.

In the k-core analysis all vertices of a connected graph belong to the 1-core. In figure 7.2 this is indicated by the largest line encircling the entire graph. Then, all vertices of degree $d < 2$ are recursively cut out. In Figure 7.2 these are the blue vertices and they constitute the 1-shell. All other vertices maintain a degree of $d \geq 2$ after pruning the blue vertices, and are not eliminated in this step. The remaining vertices form the 2-core, enclosed by another dotted line. In the next step vertices with degree $d < 3$ are pruned revealing the 3-core. Note that $c_{max} = 3$ in the graph in Figure 7.2 as after pruning no vertex has a degree $d > 3$. Also note that the coloring of the vertices in the Figure indicate their shellness.

In addition to the above definitions, let us also consider that each vertex i has a label associated with it l_i and L is the set of all labels found in the graph G and s is the number of distinct labels found in L.

Layout Equations

The k-pie visualization algorithm positions vertices in 2 dimensional space. The position of each vertex depends on its shellness and its semantic label. Each vertex i is positioned using a pair of polar coordinates (ρ_i, α_i). The

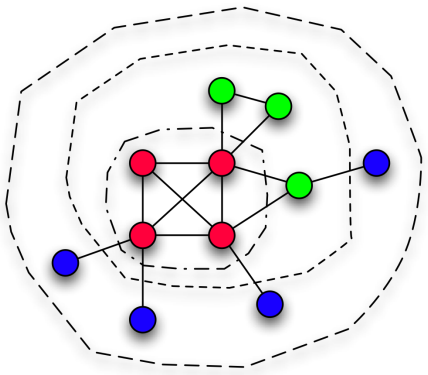


Figure 7.2: The k-core decomposition for a small graph. Each closed line contains the set of vertices belonging to a given k-core, and the color of the vertices distinguish different k-shells.

radius ρ_i depends on the shellness of i and that of its neighbors while the angle α_i depends on the label associated with i. In the resulting visualization, k-shells are presented as layers of concentric circles with the innermost circle corresponding to the vertices with the highest shellness. Then vertices sharing the same label are positioned within in a certain angular range. This creates pie-like 'slices' of vertices sharing the same label across the k-shell layers. Hence the name of the algorithm *k-pie*.

The calculation of ρ_i is as follows:

$$\rho_i = (1-\epsilon)(c_{max} - c_i) + \frac{\epsilon}{|V_{c_j} \geq c_i(i)|} \sum_{j \in V_{c_j} \geq c_i(i)} (c_{max} - c_i) \tag{7.1}$$

where $V_{c_j} \geq c_i$ is the set of neighbors of i having shellness c_j greater or equal to c_i. The parameter ϵ is a tuning parameter to control the possibility of rings overlapping.

Then, the angle α_i is calculated as follows:

$$\alpha_i = 2\pi \sum_{1 \leq m < l_i} \frac{|L_m|}{n} + N\left(\frac{|L_{l_i}|}{2n}, \frac{2\pi|L_{l_i}|}{n}\right) \tag{7.2}$$

where L is an ordered list of the labels in the graph and $|L_m|$ is the number of vertices with the label m, N is a normal distribution of mean $\frac{|L_{l_i}|}{2n}$ and width $\frac{2\pi|L_{l_i}|}{n}$. Assuming L is an ordered list of labels, referring to $m < l_i$ allows us to allocate the appropriate portion of the angular space to a given label.

Color and Size of Vertices and Edges

The size of each vertex in the visualization corresponds to the logarithm of its degree. Generally, the larger (higher degree) vertices be nearest the center of the visualization although it is possible for a higher degree vertex to exist in one of the more outer shells - this is especially true when the network does *not* exhibit assortative mixing with respect to degree (see section 2.1.7 for an explanation of assortativity).

The coloring of the vertices is related to the label associated with each vertex. Each distinct label found in L is given a unique color.

The drawing of edges can be considered optional. In fact drawing all the edges in a larger graph will result in an unintelligible tangle. With no edges drawn at all, the resulting visualization is still useful. A homogeneously randomly sampled fraction of edges can be drawn to help convey the sense that the visualization is of a network, not just some grouping of nodes. This approach does not add to computational cost significantly and is used in [Alvarez-Hamelin et al., 2006]. A more computationally expensive approach is to only draw edges with a higher *betweenness centrality* - those edges which are found more often in the shortest paths between a pair of vertices [Freeman, 1977].

Complexity

The k-pie algorithm has a complexity that is nearly identical to that of k-core decomposition. If we assume no re-ordering of L we can index our list of labels for the angular calculation in $\mathcal{O}(s * n)$ where s is the number of labels. Generally s will be small compared to n the number of vertices. The k-core decomposition takes time $\mathcal{O}(n + e)$ - $\mathcal{O}(n)$ to build a list of vertex's degree and $\mathcal{O}(e)$ to perform the pruning in the recursive decomposition step where e is the number of edges. So our total time complexity for k-pie is $\mathcal{O}(s*n+n+e)$ or simply $\mathcal{O}(n+e)$ if the number of distinct labels s is small.

7.2.2 K-Pie Music Artist Browser

The k-pie algorithm could be applied to most any graph-like data that includes vertex labels of some kind. Here we will discuss the application that motivated our development of k-pie - visualization of the MySpace artist network. This application was created as a demo and was never released and is not currently maintained.

As discussed in section 3.6 the MySpace artist network consists of a subset of the MySpace social network that includes only users who specify that they are "artists". A network is constructed from the directed "top friend" relationship between pairs of artists. This relation is chosen over the general undirected "friend" relation because it is assumed to be more meaningful and salient in terms of musical similarity. That is to say, a "top friend" relationship is more likely to imply some musical relationship between two artists - collaboration, co-membership, or stylistic influence.

Figure 7.3: k-pie layout visualization of a sample of the MySpace artist network where the slices and vertex colors correspond to genre labels. Note that genre label text is only printed for genre labels that constitute $> 1\%$ of vertex labels.

We can see the results of the k-pie visualization of this data set in Figure 7.3. Each vertex represents a music artist and the color of each vertex indicates the primary genre label associated with that artist. This data set contains $n = 15,019$ vertices (music artists) and $e = 114,606$ edges (top friend relations) and $s = 106$ labels (genre). Note in this visualization we have opted not to draw edges at all as a stylistic choice. Notice that a few highly connected music artists gravitate towards the center belonging to the

highest k-shells around $c_{max} = 29$. However the shells immediately lower than c_{max} are mostly empty. This behavior is indicative of *scale-free* networks where the cumulative degree distribution follows a power law decay - $P_c(d) \sim d^{-(\alpha-1)}$ Newman [2003b]. Also note that the highly connected vertices in the center k-shells constitute a "rich club" where the music artists with the highest degree values are also connected to each other Costa et al. [2007]. We can also get a sense for what musical genres dominate this sample of the MySpace artist network. We can see that "Hip Hop" (yellow) and "Rap" (bright green) account for nearly half of all the genre labels in the data set. We also see that the rich club in the center of the visualization includes vertices with genre labels "Hip Hop", "Rap", "Soul", "Reggae", and "Hardcore" - a somewhat surprising addition to the list. Note that the genre label appears as text only for those genre labels that are associated with more than 1% of the vertices in the network. The data set actually includes 106 unique genre labels and therefore the visualization contains 106 distinct colors. However, it can be exceedingly difficult for the viewer to accurately distinguish between so many colors therefore text labels are used in favor of a color legend. The vertices are drawn to be translucent so the viewer can get a better sense vertex concentration where vertices fall on top of one another.

7.3 Public Domain Classical Recordings

As a proof-of-concept implementation for the similarity framework described in chapter 5, we collect a small set of public domain classical music recordings and some associated metadata, apply some similarity derivation methods, and publish the data set in a SPARQL endpoint.[4]

The audio recordings were collected from the Musopen[5] website which curates a collection of public domain recordings of classical music. Unfortunately, the metadata associated with these recordings was not very consistent. Determining who composed the piece, who performed the piece, and who conducted the piece was not straight-forward. For this application we were primarily concerned with associating composers with audio recordings and side-stepping the FRBR details of accurate Music Ontology modeling. For this reason we introduce the `ov:composer` property which associates a `mo:Track` with artist who is credited with the composition. The noisy and incomplete metadata of the Musopen recordings was matched against the list of classical composers from the Classical Music Navigator website dis-

[4] see http://classical.catfishsmooth.net/about/

[5] see http://musopen.org/

cussed in section 3.4. These composers are in turn matched with appropriate resources in the MusicBrainz database.

The CMN resource already provides us with influence relationships between composers. We model these relationships as blank nodes of type `sim:Association` and bind these associations to a URI[6] we mint ourselves which identifies the Classical Music Navigator influence association method.

We apply two additional association methods that are based on audio signal analysis. Audio signal analysis was performed on the audio files using the Sonic Annotator software[7]. Sonic Annotator allows for audio files to be batch processed with any Vamp audio analysis plugin[8]. With Sonic Annotator, the signal analysis results can be output in RDF [Cannam et al., 2010].

Lightweight timbre features are calculated based on the means and variances of the audio signals' mel-frequency cepstral coefficients (MFCCs). A modified Kullback-Lieber divergence calculation is used to calculate an inter-song similarity measure as described in [Levy and Sandler, 2006]. This comprises another `sim:AssociationMethod` for which we mint another URI.[9] Dereferencing this we see the `sim:description` property pointing to an N3-Tr graph that uses, among other things, Vamp plugin URIs to disclose the similarity derivation workflow.

Because our data set is relatively small (800 audio files) we perform an exhaustive comparison but only record similarities where the timbral distance is below a give threshold.

In addition to timbre features, we use audio analysis to estimate the musical key of the song. This is done following the tonality estimation approach outlined in [Noland and Sandler, 2007]. This approach results in a sequence of key estimations across the entire song. We naively assume that the key detected most frequently can be selected as the "global" key of the song. We consider songs similar if they have the same global key giving us a second `sim:AssociationMethod` which we assign a URI.[10] Here we also use the `sim:description` property to point to an N3-Tr graph disclosing the workflow.

[6] the CMN influence association is identified by http://classical.catfishsmooth.net/resource/cmn-influence

[7] http://www.omras2.org/SonicAnnotator

[8] http://www.vamp-plugins.org/

[9] the timbre similarity association method is identified by http://classical.catfishsmooth.net/resource/mv-timbre-sim

[10] the audio-based key analysis association is identified by http://classical.catfishsmooth.net/resource/same-key-sim

With these three `sim:AssociationMethods` we can make some rather interesting queries. In listing 7.1 we combine all three notions of similarity into one query.

```
SELECT DISTINCT ?dist ?composer_name ?title ?af WHERE
{
# find similarity statements involving timbre with the seed track
?s sim:method <http://classical.catfishsmooth.net/resource/mv-timbre-sim> ;
  sim:element <http://classical.catfishsmooth.net/resource/track/362> ;
  sim:element ?track ;
  sim:distance ?dist .
?track mo:available_as ?af ;
  dc:title ?title ;
  ov:composer ?composer .
?composer foaf:name ?composer_name .

# must have same key as well
?s2 sim:method <http://classical.catfishsmooth.net/resource/same-key-sim> ;
  sim:element <http://classical.catfishsmooth.net/resource/track/362>;
  sim:element ?track .

# must have composer who influenced Wagner
<http://classical.catfishsmooth.net/resource/track/362> ov:composer ?c .

?s3 sim:method <http://classical.catfishsmooth.net/resource/cmn-influence> ;
  sim:object ?c
  sim:subject ?comp2 .
?track ov:composer ?comp2 .

# only return results with a distance less than 8.0
FILTER ( xsd:float(?dist) < "8.0"^^xsd:float ) .
}
ORDER BY ASC (?dist)
```

Listing 7.1: A query against the public domain classical music dataset involving three distinct types of similarity.

Starting with a recording of Richard Wagner's "Die Meistersinger von Nurnberg" we query for other songs that are similar in terms of timbre, in the same key, and composed by composers who have influenced Wagner. This

query and additional queries can be executed using the interactive AJAX client.[11]

7.4 MuSim and AudioDB

AudioDB[12] is a feature-vector store and query engine for similarity matching in vector spaces, developed from observations about effective methods for performing similarity search on large collections of audio and other multimedia [Casey et al., 2008; Casey and Slaney, 2006]. It is designed to be highly scalable and to operate on audio segments - allowing for more general searches than simple track-to-track matching. A probabilistic indexing scheme based on Locality Sensitive Hashing is used such that a single linear scan of the database can produce an index data structure where retrieval of similarity results scales sub-linearly with respect to the number of items in the database even for very high-dimensional spaces.

Although no part of audioDB was developed in support of this thesis, a recent addition to the software makes use of the Similarity Ontology. In an effort to incorporate the power of audioDB into the linked data ecosystem, an RDF store facade was constructed that supports SPARQL queries [Cannam et al., 2010]. Given that audioDB actually stores feature vectors and that storing $\prime(n^2)$ similarity statements would be impractical and unscalable - the audioDB SPARQL interface actually calculates similarity judgments on demand. An internal memory storage model is used as a cache to store results and temporary objects for similarity queries, with audioDB itself accessed for feature information and to perform similarity searches. The presence of an triple pattern

However, the audioDB SPARQL support does have some significant issues. In a divergence from the SPARQL recommendation, the ordering of predicates is critical in the audioDB SPARQL implementation. Note that in in listing 7.2 the `sim:element` predicates appear before the `sim:distance` predicate. This is required in the current implementation.

Furthermore, the distance querying process compares tracks on an individual basis, but for many queries (such as the one in listing 7.3), it would be possible to perform the query with a single call to audioDB. Adapting the storage module to support this form of optimization is difficult as statement templates are supplied individually, and the results are expected immediately.

[11] see http://classical.catfishsmooth.net/snorql/

[12] available at http://omras2.doc.gold.ac.uk/software/audiodb/

```
PREFIX sim: <http://purl.org/ontology/similarity/>
PREFIX ksa_charm: <http://omras2.gold.ac.uk/catalogue/ksa_charm/>

SELECT ?distance WHERE {
  _:s a sim:Similarity;
    sim:element ksa_charm:KSA_CHARM_339;
    sim:element ksa_charm:KSA_CHARM_309;
    sim:distance ?distance.
}
```

Listing 7.2: SPARQL query to retrieve the distance between two signals

This is particularly inefficient the query in listing 7.3, as every track must be compared to every other track in separate calls to the backend database.

```
PREFIX mo: <http://purl.org/ontology/mo/>
PREFIX af: <http://purl.org/ontology/af/>
PREFIX sim: <http://purl.org/ontology/similarity/>
PREFIX ksa_charm: <http://omras2.gold.ac.uk/catalogue/ksa_charm/>

SELECT ?signal ?distance WHERE {
  ?signal a mo:Signal.
    _:s a sim:Similarity;
    sim:element ksa_charm:KSA_CHARM_339;
    sim:element ?signal;
    sim:distance ?distance.
}
ORDER BY (?distance) LIMIT 5
```

Listing 7.3: SPARQL query to retrieve the 5 signals closest to the input

Also, the query must be written in a specific order to ensure that the storage model is able to perform the search. As such, `sim:Similarity` individuals must be declared prior to any elements, and element predicates must be declared prior to the distance predicate. The `sim:Similarity` predicates should be allowable in any order, but as the distance predicate relies on knowing the two signals to compare, this is currently impossible.

It should also be noted that the feature import process disregards metadata about the feature extractor and source audio. This information must be

obtained by querying against an additional metadata store. This is straightforward when using two queries with two separate SPARQL endpoints, but techniques to execute queries against multiple endpoints are not yet standardized.

Although the audioDB SPARQL implementation is not without its faults, it demonstrates how the Similarity Ontology can be applied to a vector space query engine as a sort of output layer interfacing with the linked data world.

7.5 CatfishSmooth

CatfishSmooth[13] is a web application for browsing the connections between popular music artists. Connections that are somewhat tangential to music are presented (e.g. connections related to geographic locations or religious affiliations) alongside connections that are more music-related (e.g. artists that play the same instrument).

7.5.1 Motivation

The CatfishSmooth application was motivated by a desire to apply linked data to music exploration. The aim was to show how a series of relatively simple SPARQL queries could be used to build a useful end-user application.

7.5.2 Implementation

Each music artist in the application is identified by a URI based on an appropriate MusicBrainz identifier (these identifiers are described in section 3.9.1). When a music artist's CatfishSmooth URI is dereferenced by a normal web browser a user interface similar to that shown in figure 7.4 is presented.

The top section of the interface contains media related to the given music artist. This media is collected from the YouTube API[14] and the Echonest API[15]. Then each box in the lower part of the interface represents a set of associations with other artists. The top green section of the box contains text that explains the association such as "artists that are also People From

[13] see http://catfishsmooth.net/
[14] see http://www.youtube.com/dev
[15] see http://developer.echonest.com/

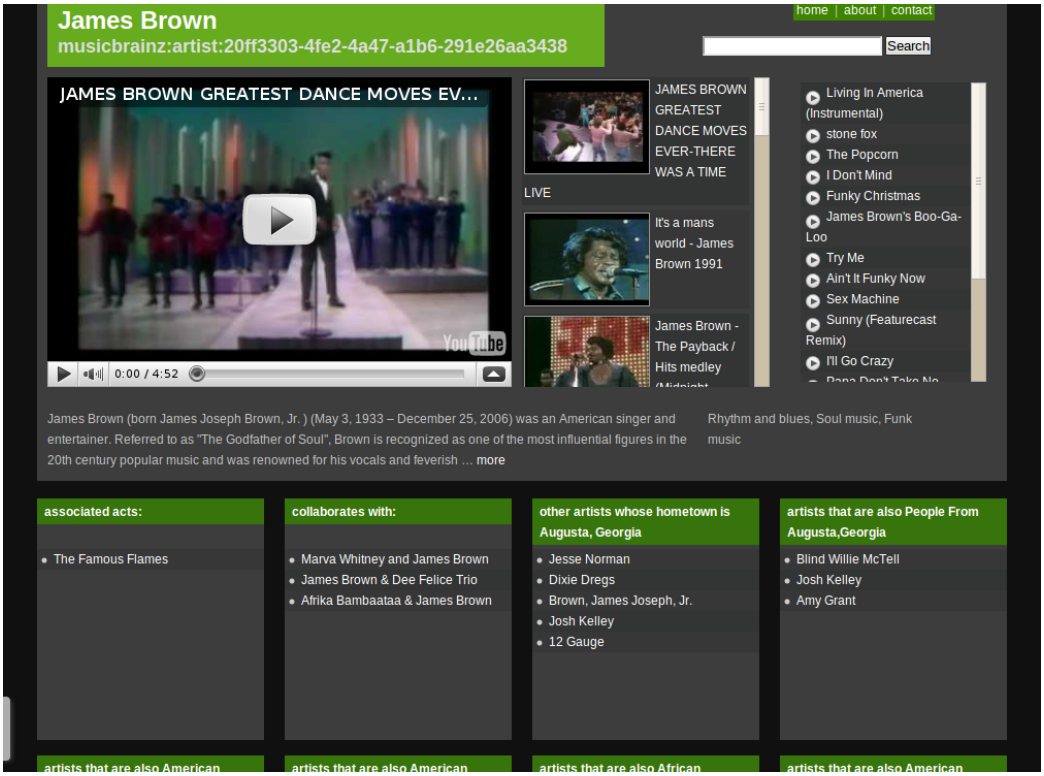


Figure 7.4: A screen shot of the CatfishSmooth web application user interface.

Memphis, Tennessee" or "artists that are also Incarcerated Celebrities". The lower gray area is a list of artists that share the given association. Each artist name in the list links to that artist's CatfishSmooth URI so the end user can continue exploring for new artists.

When an artist's webpage is loaded, data is asynchronously aggregated from various SPARQL endpoints and RDF resources including DBpedia (described in section 4.6.2), DBTune (described in section 4.6.1), and SameAs.org[16]. This means, in its current form, the CatfishSmooth web application does not require any type of database backend. All the data is stored remotely as linked data.

7.5.3 Evaluation

As a means of evaluating the CatfishSmooth website, and by proxy the similarity framework powering it, a system usability survey was preformed. The

[16] see http://sameas.org

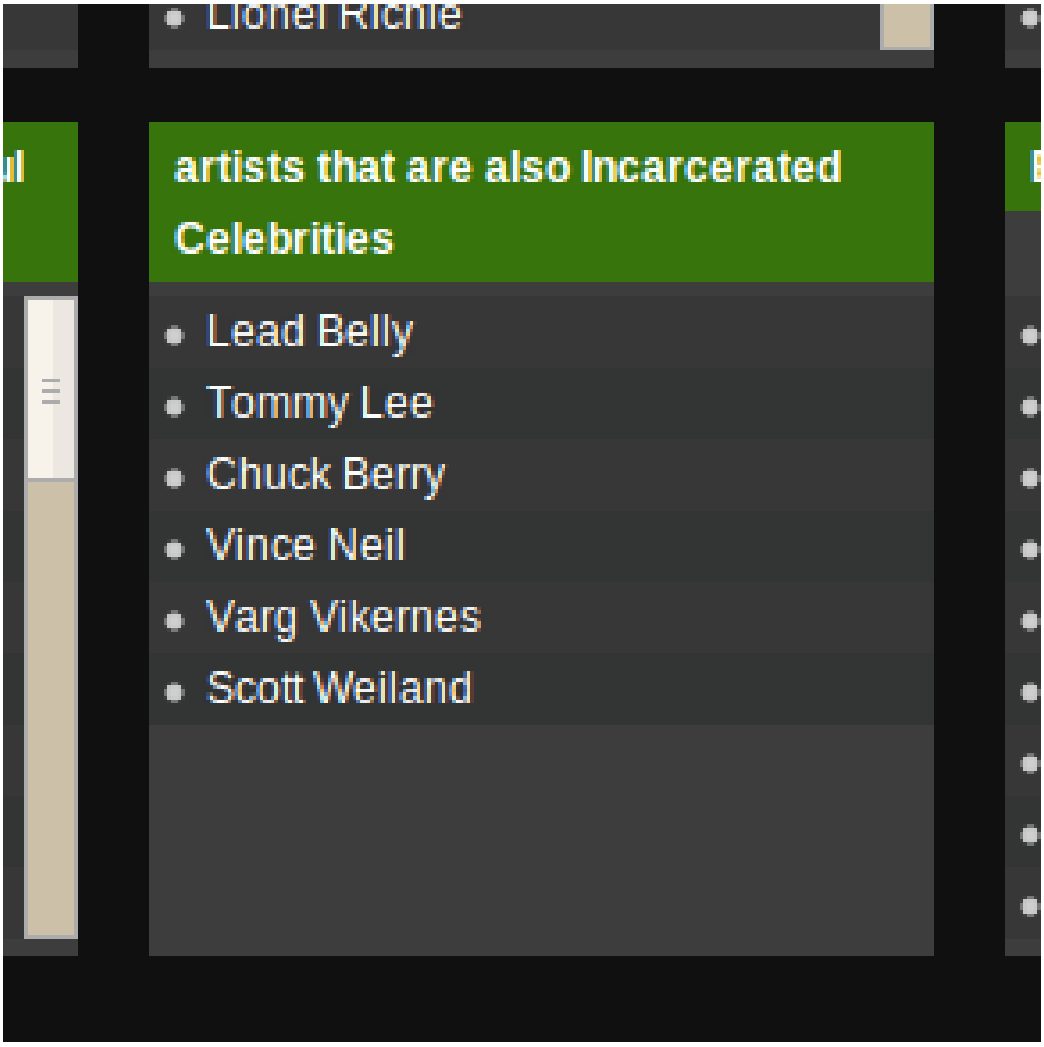


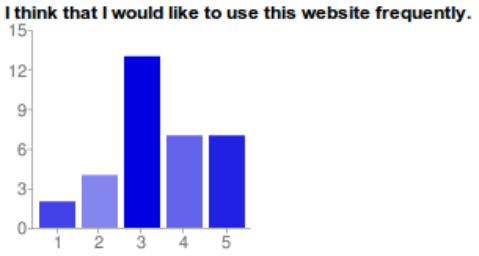
Figure 7.5: A listing of music artists who are also incarcerated celebrities.

survey consisted of 10 simple questions based on the system usability scale where users rate on a scale of 1 to 5 whether they disagree or agree with various statements about the system under consideration [Brooke, 1996; Bangor et al., 2008]. The survey was posted on the website for 30 days starting 2011-1-19. In total 33 anonymous users participated in the survey. The survey questions where as follows:

- I think that I would like to use this website frequently.
- I found this website unnecessarily complex.

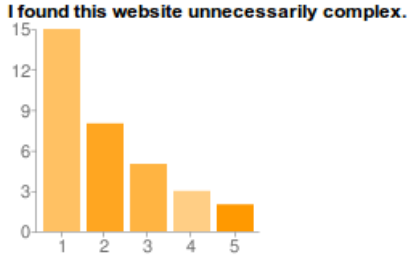
- I thought this website was easy to use.
- I think that I would need assistance to be able to use this website.
- I found the various functions in this website were well integrated.
- I thought there was too much inconsistency in this website.
- I would imagine that most people would learn to use this website very quickly.
- I found this website very cumbersome/awkward to use.
- I felt very confident using this website.
- I needed to learn a lot of things before I could get going with this website.

In accordance with the classic system usability scale design 10 statements are included and positive statements are interleaved with negative statements. The responses for the survey are shown in Figures 7.6 - 7.15.



1 -	strongly disagree	2	6%
2		4	12%
3		13	39%
4		7	21%
5 -	strongly agree	7	21%

Figure 7.6: Results for question 1 "I think that I would like to use this website frequently."

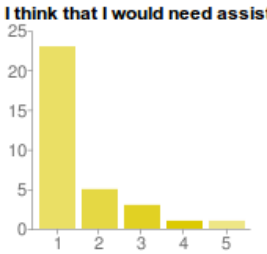


1 -	strongly disagree	15	45%
2		8	24%
3		5	15%
4		3	9%
5 -	strongly agree	2	6%

Figure 7.7: Results for question 2 "I found this website unnecessarily complex."

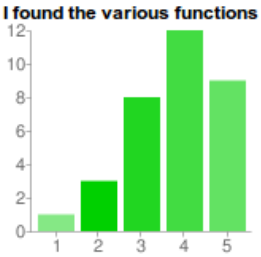
1 -	strongly disagree	1	3%
2		2	6%
3		3	9%
4		13	39%
5 -	strongly agree	14	42%

Figure 7.8: Results for question 3 "I thought this website was easy to use."



1 -	strongly disagree	23	70%
2		5	15%
3		3	9%
4		1	3%
5 -	strongly agree	1	3%

Figure 7.9: Results for question 4 "I think that I would need assistance to be able to use this website."



1 -	strongly disagree	1	3%
2		3	9%
3		8	24%
4		12	36%
5 -	strongly agree	9	27%

Figure 7.10: Results for question 5 "I found the various functions in this website were well integrated."

Figure 7.11: Results for question 6 "I thought there was too much inconsistency in this website."

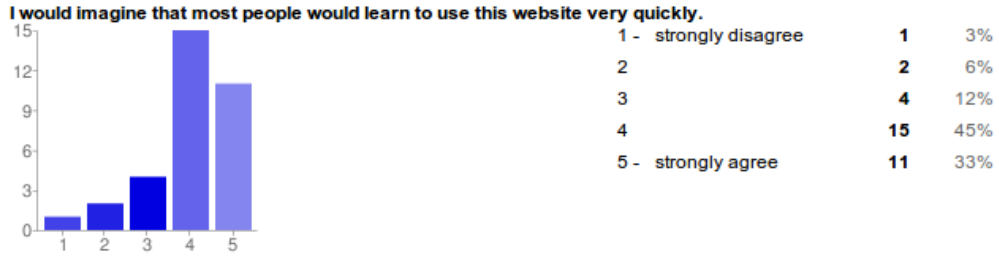


Figure 7.12: Results for question 7 "I would imagine that most people would learn to use this website very quickly."

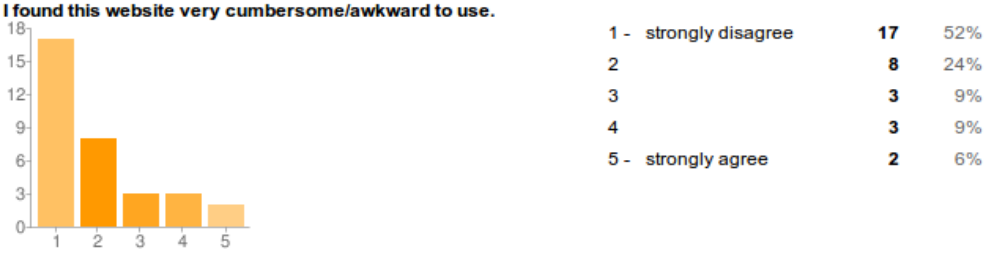


Figure 7.13: Results for question 8 "I found this website very cumbersome/awkward to use."

Figure 7.14: Results for question 9 "I felt very confident using this website."

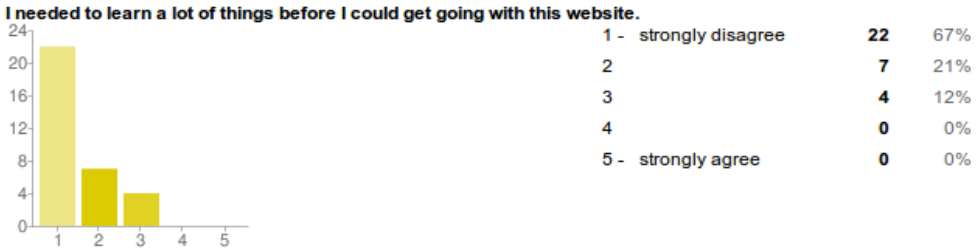


Figure 7.15: Results for question 10 "I needed to learn a lot of things before I could get going with this website."

Scores are summed following Brooke [1996]. Each item's score contribution will range from 0 to 4. For items 1,3,5,7,and 9 the score contribution is the scale position minus 1. For items 2,4,6,8 and 10, the contribution is 5 minus the scale position. We multiply the sum of the scores by 2.5 to obtain the overall value of *s*ystem *u*sability (SU) which will be in the range of 0 to 100. For the 33 respondents to our survey the average SU is 75 with a standard deviation of 18.4. This indicates a generally positive response for the usability of the CatfishSmooth application.

7.5.4 Future work for CatfishSmooth

In future incarnations of the CatfishSmooth website a triple store backend will be used. The data aggregated for each artist will be stored locally in RDF and appropriate association statements will be generated. Each association box on the interface will correspond to a `sim:Association` statement using the Similarity Ontology. An N3 graph that can be translated into the SPARQL queries or series of SPARQL queries used to generate the association box will be bound to the associations statement using as the `sim:grounding` predicate. Ultimately, end-users will be able to add their own associations between music artists with a read/write wiki-like interface.

7.6 DBTune Extensions

As mentioned in section 4.6.1, several extensions to the DBTune project were implemented as part of this thesis work.

7.6.1 Artist Similarity

The CatfishSmooth web application described in section 7.5 makes use of two artist similarity services that have been added to DBTune. These services are both wrappers around web APIs.

Last.fm Artist Similarity

The last.fm artist similarity service[17] provides the artist similarity statements from last.fm as linked data. This is a wrapper around the last.fm API[18] that

[17] see http://dbtune.org/artists/last-fm/
[18] see http://www.last.fm/api

translates artist similarities into MuSim RDF. Since last.fm uses MusicBrainz identifiers, the service makes use of MusicBrainz identifiers to mint URIs for artists. This makes inter-linking to other artist data easy - links are provided to the DBTune MusicBrainz service and BBC/music. When dereferenced the last.fm service provides a series of similarity statements involving the target artist in MuSim RDF.

Echonest Artist Similarity

The Echonest artist similarity service[19] provides very similar functionality but uses the Echonest artist similarity API[20]. Again, because Echonest makes use of MusicBrainz identifiers we use these to mint artist URIs for our service. A dereferenced URI returns a series of similarity statements involving the target artist and links to other resources at DBTune/MusicBrainz and BBC/music.

7.6.2 Classical Composers

The Classical Composer data set[21] provides an array of information about concepts and individuals related to the canon of Western Classical Music. This includes the composer influence relations from the Classical Music Navigator discussed in section 3.4 as well as data aggregated from around the web. This data set is, to some extent, hand curated by Chris Cannam and provides appropriate links to resources in DBpedia, DBTune/MusicBrainz, and BBC/Music.

7.7 LinkedBrainz Advanced Relationships

The LinkedBrainz project[22] is a JISC-funded project to provide the music metadata from the MusicBrainz project (first discussed in section 3.9) using linked data techniques (as described in section 4.6. Although previous translations of MusicBrainz data into RDF exist (see section 4.6.1) these mappings do not deal with the Next Generation Schema or Advanced Re-

[19] see http://dbtune.org/artists/echonest/
[20] see http://developer.echonest.com/
[21] see http://dbtune.org/classical/
[22] see http://linkedbrainz.c4dmpresents.org/

lationships. The Next Generation Schema or NGS[23] is a significant update to the MusicBrainz data model which, among other things, makes the distinction between music recordings (a `mo:Signal` in Music Ontology terms) and a musical work (a `mo:MusicalWork`). In this way the Next Generation Schema actually allows for better alignment of MusicBrainz entities to Music Ontology concepts. Furthermore, with the LinkedBrainz project, linked data will be served directly from the MusicBrainz servers - removing a replication step and hopefully consolidating the variants of music-related URIs based on MusicBrainz identifiers.

Although the Next Generation Schema maps readily to Music Ontology concepts the Advanced Relationships present more of a challenge. Recall that, as described in section 3.9.2, Advanced Relationships allow MusicBrainz editors to specify associations between pairs of entities in the database. For example, it is possible to specify that two artists are married to each other or that they share some other familial relationship. It is also possible to specify that a recording is a performance of a particular work or that one recording samples another recording. As of this writing, there are just over 300 Advanced Relationship types in the MusicBrainz database. Some of these relationships - for example marriage - can readily be described by properties in existing ontologies[24] however other relationships - for example "mashes up" or "booking support" - are not readily covered by existing ontologies. It has been proposed[25] that such Advanced Relationships be modeled using the MuSim ontology.

The relational database schema for Advanced Relationships in the MusicBrainz database is shown in figure 7.16.

In the RDF mapping, each entry in the `link_type` table that does not readily map to a property in some existing ontology becomes an instance of type `sim:AssociationMethod` identified by a URI constructed from the corresponding MusicBrainz identifier (each Advanced Relationship has an MBID that is stored in `link_type.gid`). The values for `link_type.entitytype0` and `link_type.entitytype1` become the `sim:domain` and `sim:range` of the `sim:AssociationMethod` assuming the relationship is directed. For the undirected case, `sim:scope` is used. The MusicBrainz Advanced Relationships schema does not clearly distinguish between directed and undirected relationships. However, a "link phrase" (`link_type.phrase`) is provided in the

[23] details about NGS can be found at http://wiki.musicbrainz.org/Next_Generation_Schema

[24] the `rel:spouseOf` property is defined in the relationship vocabulary at http://vocab.org/relationship/

[25] see proposals at http://wiki.musicbrainz.org/NGS_to_RDF_mappings

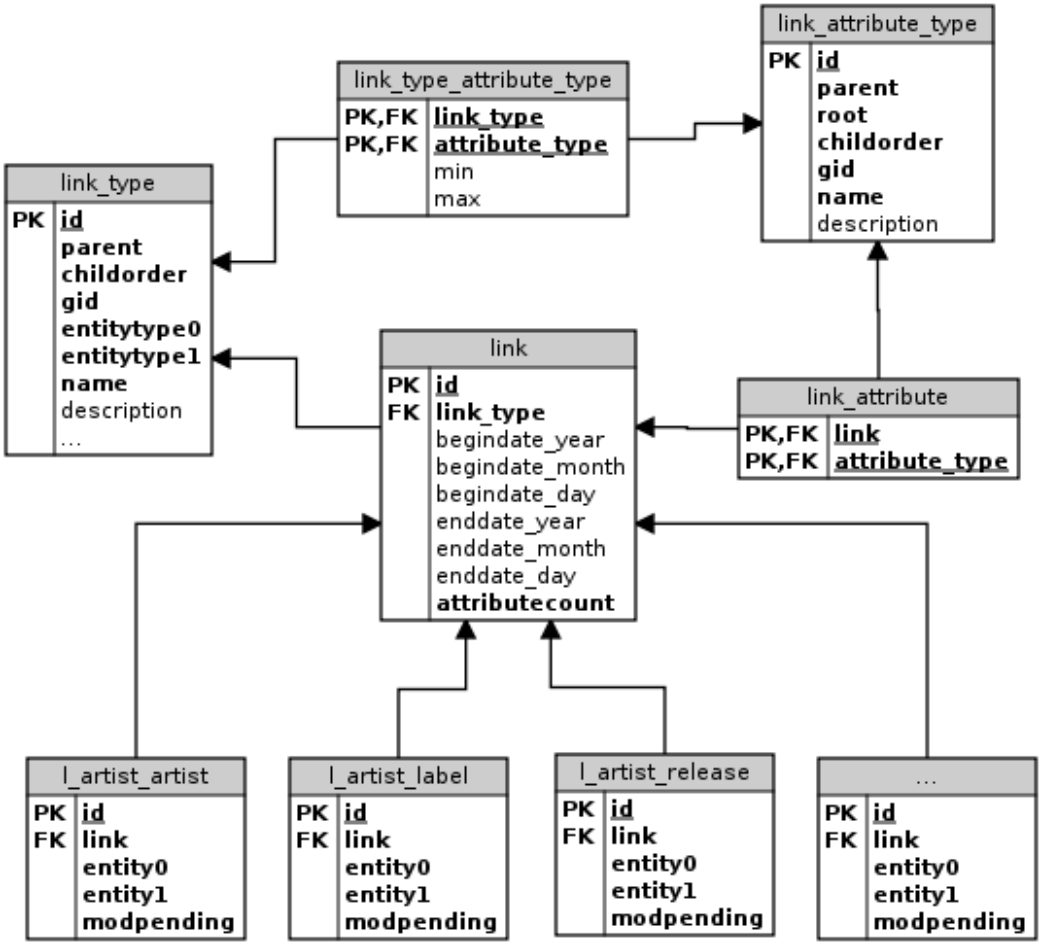


Figure 7.16: Advanced Relathionships database schema for MusicBrainz.

table along with a "reverse link phrase" (`link_type.r_phrase`) for presenting the relationships in natural language. We can assume that if the values of these columns are equivalent the relationship is undirected and otherwise the relationship is directed - although this is not an ideal solution.

To generate a set of association statements a join query is required. For example a join on the `link` table and the `l_artist_release` table returns all the associations between artists and releases. These are modeled as a `sim:Association` and bound to the appropriate method using the `sim:method` predicate. A similar approach can be applied to artist-artist relationships, artist-recording relationships, artist-label relationships etc. This approach has already been applied to create an experimental mapping that

can be queried via SPARQL[26].

Of course one migth argue that a relationship like booking support might be outside of the scope of association. This can be addressed using the `sim:AssociationMethod` modeling approach however as we describe above. If a particular application or data consumer decides this type of relationship is out of scope, they can simply exclude statements bound to these association

7.8 Summary

In this chapter we have described various applications developed as part of this thesis. We begin by describing the Classical Music Universe application in section 7.1 which uses linked data and graph layout algorithms to create a browser for discovering classical composers. In section 7.2 we describe the k-pie layout algorithm which is a graph layout algorithm for graphs with labeled nodes. In section 7.3 we describe a linked data resource for public domain classical music that leverages our similarity framework. In section 7.4 we describe the extensions to audioDB which express the results of metric space similarities using MuSim. In section 7.5 we describe the CatfishSmooth web application which is largely based on the MuSim model for associations. Additionally we provide an evaluation in the form of a system usability survey. We describe various extensions to the DBTune linked data resource in section 7.6 and finally show how the MuSim framework has been applied to the LinkedBrainz project in section 7.7.

[26] see `http://linkedbrainz.c4dmpresents.org/snorql`

Chapter 8

Conclusions

Below every tangled hierarchy lies an inviolate level.

– Douglas Hofstadter, *Gödel Escher Bach*, 1979

We have discussed connections in music. We have examined connections between music artists as they are currently found on the web and we have proposed a framework for describing inter-entity associations and similarities in a distributed fashion as part of the web of data. We have presented a method for evaluating our framework based on the models of cognitive pschology and developed several applications for our framework.

In this chapter we will summarize our work, discuss its limitations, and present ideas for future work.

8.1 Review of Contents

In order to better understand the nature of connections between music-related entities we provide a survey of music artist networks found on the web in chapter 3. A review of previous work that has analyzed artist networks including the Last.fm artist recommendation network and the various All Music Guide networks is presented. We then provide original analysis of seven distinct artist networks including the Classical Music Navigator composer influence network (section 3.4), the Discogs artist-release network (section 3.5), the MySpace artist top-friends network (section 3.6), the MySpace artist audio-based similarity network (section 3.6.4), the Soundcloud artist network (section 3.7), the Echonest artist similarity network (section

3.8), and the MusicBrainz Six Degrees artist network (section 3.9). All these networks have some characteristics in common. Most notably they can all be called "small world" networks in that they have relatively small values for average geodesic distance ($\langle d \rangle$) and diameter (d_{max}) while having a high clustering coefficient (T). Whenever the network construction methodology would allow it, we find networks that are fragmented - with one giant connected component and many smaller components. This has some strong implications for using these networks for recommendation and navigation - some artists will always be unreachable. Other aspects of artist network structure varied considerably. Some degree distributions approximate a power-law while others are closer to an exponential distribution. Some networks are assortative with respect to degree, others exhibit random mixing or slightly dissassortative mixing. In some cases the network structure has a rather clear correlation with popularity (measured in terms of play counts) and in other cases the network structure is orthogonal to popularity. In summary, applying different edge connection criteria to music artist networks can have a drastic effect on the network structures that emerge. All inter-artist connections are not created equal.

In chapter 4 we review semantic web technologies and how they have been applied to the domain of music. We review the particulars of technologies like the URI (section 4.1), RDF (section 4.2), OWL/RDFS (section 4.3, and SPARQL (section 4.5). We describe the Music Ontology developed by Raimond [2009] and how its event decomposition approach can be applied to modeling music-related knowledge. Then we show how linked data methodologies have been applied to enable a distributed information space for music-related knowledge.

In chapter 5 we develop our framework for describing inter-entity associations in this web of data. At the center is the MuSim ontology which adopts a reification approach to describing similarities and associations. Associations are treated as a *concept* rather than a *property* - allowing additional predicates to be bound to an association and enabling an intuitive and efficient querying paradigm. We describe how our framework could leverage existing provenance frameworks and how transparency can be provided through the enumeration of workflows.

Our association framework is evaluated in chapter 6. Cognitive models for similarity from the cognitive psychology literature are modeled using our framework. By combining a set of constraints expressed using Notation 3 logic and the MuSim ontology we are able to accommodate each of the four cognitive models for similarity we explore. Additional qualitative evaluation is provided by enumerating a series of use-case scenarios and describing how

our framework can accommodate these scenarios.

Finally we present a series of applications developed as offshoots of this thesis work. Some applications are enabled by our work in artist network analysis, others are enabled by the web of linked data, and still others leverage our association framework.

8.1.1 Satisfied Requirements

In section 1.3 we outlined a set of requirements for the associations framework we develop in this thesis. Here we will review this list and discuss how we have satisfied these requirements.

- **Heterogeneous** - A wide variety of similarities and associations related to music can already be found on the web. We surveyed networks of inter-artist connections found on the web in chapter 3 and demonstrated the diversity of these network structures. In section 6.4 we applied our framework to modeling these various artist networks and provide a proof-of-concept knowledge base.

- **Expressive** - Similarity and associations are often complex and multifaceted. Any pair of entities potentially have an infinite number of distinct associations between them. Our framework accommodates multifaceted similarity by allowing for multiple inter-entity associations based on distinct association methods as demonstrated in section 6.2.3.

- **Explicit** - The meaning of our associations should be explicit to both human users and computers and have precise semantics. We achieve this through the use of the RDF/OWL technology stack with the `sim:AssociationMethod` concept being the focal point of our semantic model as developed in section 5.2.3.

- **Auditable** - Some connections are grounded in undisputtable facts while other connections are grounded in opinion while still others are the result of some algorithmic recommendation process. In our framework we provide a mechanism for provenance as described in section 5.2.4 and for workflow transparency as described in section 5.2.5.

- **Distributed** - Our framework is intended to be an extension of the web and as such it must allow for a distributed information space that joins multiple data sources hosted in multiple places. By using linked data technologies for our framework we allow for such distribution and

we describe our vision for a similarity ecosystem in section 5.3 while providing concrete implementations in sections 6.4, 7.3, 7.6.1, and 7.7.

- **Queryable** - We must be able to ask questions about associations in our framework and compute answers in a reasonable amount of time. This achieved with the SPARQL query language as described in section 5.3.1.

8.2 Limitations and Future Work

Of course, the work presented here is not without its limitations and there are plenty of opportunities for future work.

8.2.1 Network Analysis

Our work analyzing music-related connections on the web only addresses artist-to-artist connections. Obviously a much wider variety of node types are possible when discussing networks of music-related entities. For example we could consider connections between recordings and other recordings or connections between artists and listeners. Some analysis of bipartite artist-listener networks [Lambiotte and Ausloos, 2005] and bipartite artist-playlist networks [Cano and Koppenberger, 2004; Baccigalupo and Plaza, 2007] has been performed but these studies focus on one source of information rather than providing a broad survey of information sources available on the web.

The MusicBrainz data was not analyzed directly in section 3.9. Instead an artist network used for the Six Degrees web application that was constructed using MusicBrainz data was analyzed. Future work warrants a more in-depth examination of the connections found in the MusicBrainz database. This will hopefully follow from the LinkedBrainz project discussed in section 7.7.

Also it should be noted that our work does not deal with network dynamics. We take a snapshot of the artist networks in time and analyze that snapshot - a static view of the network. Network structures evolve over time and our work does not address this evolution. Future work should address this issue by examining variances in multiple static snapshots of the same network taken over some period of time. Then, this analysis should be repeated for several distinct artist networks - as we have done for the static case.

Community detection in music artist networks is not addressed in this thesis but has been addressed in other work [Teitelbaum et al., 2008; Jacobson et al., 2008b; Lambiotte and Ausloos, 2006; Gleiser and Danon, 2003]. Applying community detection analysis to the artist network datasets presented in this work is left to future work.

8.2.2 Parsing Workflows

In our proposed framework for describing inter-entity associations using semantic web technologies we deal with the concept of workflows. The N3 model and the N3-Tr extension are used in this work for modeling workflows. However, our association framework does not mandate that these are the only options for modeling workflows. One reason is that there is a dearth of tools for parsing N3 and N3-Tr. While very many software libraries support the parsing of the Turtle RDF syntax, only a very few support full N3 parsing. Some tools that do support full N3 parsing include the Closed World Machine [Berners-Lee and Connolly, 2000] which supports inferencing based on N3 rules and Henry (described in section 4.6.1 and developed by Raimond [2009]) which is a proof-of-concept N3-Tr agent for processing music-related workflows. These tools could be modified to act as MuSim agents in a music similarity ecosystem as described in section 5.3 but as of this writing, this has not been implemented and is left as one of the most pressing items for future work.

8.2.3 Ontological Extensions

Although implementations of MuSim workflow processing tools are lacking, ontological developers have already begun to extend the Similarity Ontology to enable some specific applications. Thomas Gängler has created the Associations Ontology[1] and the Recommendation Ontology[2] in an effort to enable a specific music recommendation application. Both ontologies subclass concepts from MuSim and use the same reification approach to modeling associations.

[1] see http://purl.org/ontology/ao/associationontology.html
[2] see http://purl.org/ontology/rec/recommendationontology.html

8.2.4 Visual Interfaces

As part of this thesis we have developed two distinct visual interfaces for exploring music-related connections. The Classical Music Universe described in section 7.1 leverages linked data and complex network layout algorithms to present composer influence connections to the end user in a novel way. The MySpace music artist explorer described in section 7.2.2 provides a very similar functionality while implementing the k-pie semantic graph layout algorithm. However, these interfaces have not been evaluated. An evaluation of these interfaces and the construction of additional visual interfaces and faceted browsing interfaces for music-related linked data are left to future work.

8.3 Summary

We have explored the variety of inter-artist connections found on the web and provided some insights into the implications of the this heterogeneity. Semantic web technologies have been applied to the creation of a framework for modeling association and similarity. Motivated by the diversity of inter-artist connections found on the web, this framework embraces the complexity of inter-entity relationships while providing precise semantics that enable some interesting applications. We have shown how this framework is compatible with the models for similarity found in cognitive psychology and we have developed a series of applications that leverage this framework. Finally we have discussed the limitations of our present work and opportunities for future work.

Whether or not the framework developed here will enjoy broad enough uptake to realize the similarity ecosystem described in section 5.3 remains to be seen. Convincing music recommendation providers to agree on using this or any common model is a difficult proposition. Furthermore, such a proposition is contingent on broader uptake of semantic web technologies in general - a process that has been slower than expected thus far. Regardless, it is our hope that this work provides some useful insights into inter-entity connections in music and has convinced the reader that similarities and associations are most accurately modeled as compound things rather than one-dimensional properties.

Appendix A

Namespaces

The following namespaces are used throughout this work:

```
@prefix mo: <http://purl.org/ontology/mo/> .
@prefix sim: <http://purl.org/ontology/similarity/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix dc: <http://purl.org/dc/terms/> .
@prefix math: <http://www.w3.org/2000/10/swap/math#> .
@prefix log: <http://www.w3.org/2000/10/swap/log#> .
@prefix sig: <http://purl.org/ontology/signal/> .
@prefix ctr: <http://purl.org/ontology/ctr/> .
@prefix dbpedia: <http://dbpedia.org/resource/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
```

Listing A.1: The set of prefixes that are assumed in the turtle and N3 listings throughout this work.

Appendix B

Artist Networks as MuSim

Here we provide a turtle listing that models the music artist networks described in chapter 3 using the MuSim ontology developed in chapter 5. Only one edge from each network is modeled from brevity. The complete modeling can be found at http://dbtune.org/musim-nets/.

```
# prefixes in Appendix A are assumed
@base <http://dbtune.org/artists/nets/resource/> .

# myspace top friend network
:myspace_top_friend_assoc_meth a sim:AssociationMethod ;
  foaf:page <http://myspace.com> ;
  sim:domain mo:MusicArtist ;
  sim:range mo:MusicArtist ;
  dc:description """Artists i and j share a directed association
                    (i,j) if i specifies j as a 'top friend' in
                    the MySpace artist network.""" .

# randomly selected edge
_:b00 a sim:Association ;
  sim:subject <http://dbtune.org/myspace/uid/72415440> .
  sim:object <http://dbtune.org/myspace/uid/64991186> .
  sim:method :myspace_top_friend_assoc_meth .

# myspace audio-based similarity network
:myspace_audio-based_assoc_meth a sim:AssociationMethod ;
  foaf:page <http://myspace.com> ;
  foaf:page <http://marsyas.info> ;
```

```
  sim:scope mo:MusicArtist ;
  dc:description """Artists are associated if their most-played
                    tracks have an audio-based distance of less
                    than 0.395 as measured using the MARSYAS tools
                    on the MySpace artist network.""" .

# randomly selected edge
_:b07 a sim:Similarity ;
  sim:element <http://dbtune.org/myspace/uid/203349310> ;
  sim:element <http://dbtune.org/myspace/uid/63097726> ;
  sim:distance "0.264"^^xsd:float ;
  sim:mehtod :myspace_audio-based_assoc_meth .


# echonest artist recommendation network
:echonest_assoc_meth a sim:AssociationMethod ;
  dc:source <http://sites.google.com/site/musicviz2/ArtistSimilarityDat.tar.gz> ;
  foaf:page <http://echonest.com> ;
  sim:domain mo:MusicArtist ;
  sim:range mo:MusicArtist ;
  dc:description """Artists are associated if they are listed as
                    similar by the Echonest artist recommendation
                    network sample.""" .

# randomly selected Echonest edge
_:b01 a sim:Association ;
  sim:subject <http://dbtune.org/artists/echonest/AR/AR2W5931187FB3602C> .
  sim:object <http://dbtune.org/artists/echonest/AR/AR9UCLS1187B9B9D35> .
  sim:method :echonest_assoc_meth .

# Classical Music Navigator
:cmn_influence_assoc_meth a sim:AssociationMethod ;
  foaf:page <http://people.wku.edu/charles.smith/music/index2.htm> ;
  sim:domain mo:MusicArtist ;
  sim:range mo:MusicArtist ;
  dc:description """Artists share a directed association if it has
                    been specified that there exists an influence
                    connection between them on the Classical Music
                    Navigator website""" .

# randomly selected CMN edge
```

```
_:b10 a sim:Association ;
  sim:subject <http://dbtune.org/classical/resource/composer/nielsen_carl> ;
  sim:object <http://dbtune.org/classical/resource/composer/holmboe_vagn> ;
  sim:method :cmn_influence_assoc_meth .

# Discogs artist-release
:discogs_artist_release_assoc_meth a sim:AssociationMethod ;
  foaf:page <http://discogs.com/> ;
  sim:scope mo:MusicArtist ;
  dc:description """Artists share an undirected association if they have
                    appeared together on a release in Discogs and the
                    weight of that association is equivalent to the number
                    of shared releases.""" .

# randomly selected Discogs edge
_:b11 a sim:Association ;
  sim:element <http://discogs.dataincubator.org/artist/dj-damage> ;
  sim:element <http://discogs.dataincubator.org/artist/justina-curtis> ;
  sim:weight "2"^^xsd:int ;
  sim:method :discogs_artist_release_assoc_meth .

# Soundcould artist-release
:soundcloud_assoc_meth a sim:AssociationMethod ;
  foaf:page <http://soundcloud.com/> ;
  sim:domain mo:MusicArtist ;
  sim:range mo:MusicArtist ;
  dc:description """Artists share a directed association if one artist
                    is 'following' the other artist.""" .

# randomly selected Soundcloud edge
_:b12 a sim:Association ;
  sim:subject <http://api.soundcloud.com/users/235> ;
  sim:object <http://api.soundcloud.com/users/34725> ;
  sim:method :soundcloud_assoc_meth .

# MusicBrainz 6 degrees associations
:six_degrees_assoc_meth a sim:AssociationMethod ;
  foaf:page <http://labs.echonest.com/SixDegrees/> ;
  foaf:page <http://musicbrainz.org/> ;
  sim:domain mo:MusicArtist ;
  sim:range mo:MusicArtist ;
```

```
  dc:description  """Artists share a directed association based on
                     the relationships specified in the Six Degrees
                     of Black Sabbath web application based on data
                     from MusicBrainz.""" .

_:b13 a sim:Association ;
  sim:subject mba:b84c9483-ae0d-4f14-86ff-04a72ceadf7b ;
  sim:object mba:99002f2d-a7e4-49f4-824a-562cdb47a935 ;
  sim:method :six_degrees_assoc_meth ;
  rdfs:label "member of band" .
```

Listing B.1: The set of prefixes that are assumed in the turtle and N3 listings throughout this work.

Appendix C

The Similarity Ontology Document

This is the Similarity Ontology document in turtle. The most up-to-date version of the Similarity Ontology (aka MuSim) can be found at http://purl.org/ontology/similarity/

```
@prefix sim: <http://purl.org/ontology/similarity/> .
@prefix owl2xml: <http://www.w3.org/2006/12/owl2-xml#> .
@prefix dc: <http://purl.org/dc/terms/> .
@prefix wordnet: <http://xmlns.com/wordnet/1.6/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix mo: <http://purl.org/ontology/mo/> .
@prefix vs: <http://www.w3.org/2003/06/sw-vocab-status/ns#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix log: <http://www.w3.org/2000/10/swap/log#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

sim: rdf:type owl:Ontology ;
        dc:title "The Similarity Ontology" ;
        rdfs:comment """This is an ontology to express associations
                        between entities whether artists, tracks,
                        albums, compositional styles, sections of
                        tracks, playing techniques or anything. It
                        is designed with the hope of being easily
```

```
                        extensible, extremely flexible, and still
                        computationally reasonable."""@en ;
        owl:versionInfo "Revision: 0.2.02" ;
        vs:term_status "testing" ;
        dc:creator <http://kurtisrandom.com/foaf.rdf#kurtjx> ,
                <http://moustaki.org/foaf.rdf#moustaki> .


##################
# class concepts #
##################

sim:Association rdf:type rdfs:Class , owl:Class ;
        rdfs:comment """An abstract class to define some association
                        between things. Entities share an association
                        if they are somehow inter-connected. Generally
                        a directed association should have at lease one
                        sim:subject property and one sim:object property
                        or an undirected association should have at least
                        two sim:element properties."""@en ;
        rdfs:label "Association" ;
        rdfs:subClassOf owl:Thing ;
        rdfs:isDefinedBy sim: ;
        vs:term_status "testing" ;
        owl:equivalentClass [a owl:Restriction ;
                             owl:onProperty sim:method ;
                             owl:maxCardinality 1 ] ;

        # must have either sim:subject/object or sim:element
        owl:unionOf (
            [a owl:Restriction;
             owl:onProperty sim:elemet;
             owl:minCardinality "2"^^xsd:nonNegativeInteger ]
            [ owl:intersectionOf
              ( [a owl:Restriction;
                 owl:onProperty sim:subject;
                 owl:minCardinality "1"^^xsd:nonNegativeInteger ]
                [a owl:Restriction;
                 owl:onProperty sim:object;
                 owl:minCardinality "1"^^xsd:nonNegativeInteger ] ) ]
                    ).
```

```
sim:AssociationMethod a rdfs:Class , owl:Class ;
        rdfs:comment """A concept for representing the method used to
                        derive association or similarity statements."""@en ;
        rdfs:label "Association Method" ;
        rdfs:subClassOf owl:Thing ;
        rdfs:isDefinedBy sim: ;
        vs:term_status "testing" .

sim:Influence a rdfs:Class , owl:Class ;
        rdfs:comment """An abstract class indicating a directed
                        association of influence where the subject
                        entity has influenced the object entity."""@en ;
        rdfs:label "Influence" ;
        rdfs:subClassOf sim:Association ;
        rdfs:isDefinedBy sim: ;
        vs:term_status "testing" .

sim:Similarity a rdfs:Class , owl:Class ;
        rdfs:comment """An abstract class to define similarity between
                        two or more things.
Entities share a similarity
                        if they share some common characteristics of
                        interest. A similarity is a special type of
                        association."""@en ;
        rdfs:label "Similarity" ;
        rdfs:subClassOf sim:Association ;
        rdfs:isDefinedBy sim: ;
        vs:term_status "testing" .

sim:Network a rdfs:Class , owl:Class ;
        rdfs:comment """A network is a grouping of sim:Associations. The
                        associations that comprise a network are specified
                        using a series of sim:edge predicates."""@en ;
        rdfs:isDefinedBy sim: ;
        rdfs:label "Network" ;
        rdfs:subClassOf owl:Thing ;
        vs:term_status "testing" .


####################
```

```
# object properties #
######################

sim:domain a owl:ObjectProperty ;
      rdfs:comment """Specifies appropriate object types for the
                      sim:subject predicate for sim:Associations
                      bound to the given sim:AssociationMethod. The
                      presence of this predicate implies the given
                      sim:AssociationMethod begets directed
                      associations.""" ;
      rdfs:domain sim:AssociationMethod ;
      rdfs:range owl:Thing ;
      rdfs:isDefinedBy sim: ;
      rdfs:label "domain" ;
      vs:term_status "testing" .

sim:range a owl:ObjectProperty ;
      rdfs:comment """Specifies appropriate object types for the
                      sim:object predicate for sim:Associations
                      bound to the given sim:AssociationMethod.
                      The presence of this predicate implies the
                      given sim:AssociationMethod begets directed
                      associations.""" ;
      rdfs:domain sim:AssociationMethod ;
      rdfs:range owl:Thing ;
      rdfs:isDefinedBy sim: ;
      rdfs:label "domain" ;
      vs:term_status "testing" .

sim:scope a owl:ObjectProperty ;
      rdfs:comment """Specifies appropriate object types for the
                      sim:element predicate for sim:Associations
                      bound to the given sim:AssociationMethod.
                      The presence of this predicate implies the
                      given sim:AssociationMethod begets undirected
                      associations.""" ;
      rdfs:domain sim:AssociationMethod ;
      rdfs:range owl:Thing ;
      rdfs:isDefinedBy sim: ;
      rdfs:label "domain" ;
      vs:term_status "testing".
```

```
sim:edge a owl:ObjectProperty ;
      rdfs:comment "Specifies an edge in a sim:Network"@en ;
      rdfs:domain sim:Network ;
      rdfs:range sim:Association ;
      rdfs:isDefinedBy sim: ;
      rdfs:label "edge" ;
      vs:term_status "testing" .

sim:association a owl:ObjectProperty ;
      rdfs:comment "Binds a sim:Association to an arbitrary thing."@en ;
      rdfs:domain owl:Thing ;
      rdfs:isDefinedBy sim: ;
      rdfs:label "association" ;
      rdfs:range sim:Association ;
      vs:term_status "testing" .

sim:description a owl:ObjectProperty ;
        rdfs:comment """Specifies some description that discloses
                        the process or set of processes used to
                        derive association statements for the given
                        sim:AssociationMethod. This property is
                        depricated in favor of the more appropriately
                        named sim:workflow property."""@en ;
        rdfs:domain sim:AssociationMethod ;
        rdfs:label "description" ;
        rdfs:isDefinedBy sim: ;
        owl:equivalentProperty sim:workflow ;
        vs:term_status "depricated" .

sim:workflow a owl:ObjectProperty ;
        rdfs:comment """Specifies some description that discloses
                        the process or set of processes used to
                        derive association statements for the given
                        sim:AssociationMethod."""@en ;
        rdfs:domain sim:AssociationMethod ;
        rdfs:label "workflow" ;
        rdfs:isDefinedBy sim: ;
        vs:term_status "testing" .
```

```
sim:element a owl:ObjectProperty ;
        rdfs:comment """Specifies an entity involved in the given
                        sim:Association and implies the given
                        association is undirected."""@en ;
        rdfs:domain sim:Association ;
        rdfs:label "element" ;
        rdfs:isDefinedBy sim: ;
        vs:term_status "testing" .

sim:grounding a owl:ObjectProperty ;
        rdfs:comment """Binds an sim:Association statement directly
                        to an instantiated N3-Tr formulae or some
                        other workflow graph which enabled
                        the association derivation."""@en ;
        rdfs:domain sim:Association ;
        rdfs:label "grounding" ;
        rdfs:isDefinedBy sim: ;
        vs:term_status "testing" .

sim:method a owl:ObjectProperty ;
        rdfs:comment """Specifies the sim:AssociationMethod used to
                        derive a particular sim:Association statement.
                        This should be used when the process for
                        deriving association statements can be
                        described further."""@en ;
        rdfs:domain sim:Association ;
        rdfs:label "method" ;
        rdfs:range sim:AssociationMethod ;
        rdfs:isDefinedBy sim: ;
        vs:term_status "testing" .

sim:object a owl:ObjectProperty ;
        rdfs:comment "Specifies the object of a sim:Association implying a
                      directed association where \"subject is associated to
                      object\" but the reverse association does not necessarily
                      exist, and if it does exist, it is not an equivalent
                      association."@en ;
        rdfs:domain sim:Association ;
        rdfs:label "object" ;
        rdfs:subPropertyOf sim:element ;
        rdfs:isDefinedBy sim: ;
```

```
        vs:term_status "testing" .

sim:subject a owl:ObjectProperty ;
        rdfs:comment "Specifies the subject of an sim:Association implying a
                      directed association where \"subject is associated to
                      object\" but the reverse association does not necessarily
                      exist, and if it does exist, it is not an equivalent
                      association."@en ;
        rdfs:domain sim:Association ;
        rdfs:label "subject" ;
        rdfs:subPropertyOf sim:element ;
        rdfs:isDefinedBy sim: ;
        vs:term_status "testing" .

####################
# data properties #
####################
sim:weight a owl:DatatypeProperty ;
        rdfs:comment "A weighting value bound to a sim:Association where a
                      value of 0 implies two elements are not at all associated
                      and a higher value implies a closer association."@en ;
        rdfs:domain sim:Association ;
        rdfs:label "weight" ;
        rdfs:isDefinedBy sim: ;
        vs:term_status "testing" .

sim:distance a owl:DatatypeProperty ;
        rdfs:comment "A weighting value for an Association where a value of 0
                      implies two elements are the same individual."@en ;
        rdfs:domain sim:Association ;
        rdfs:label "distance" ;
        rdfs:isDefinedBy sim: ;
        vs:term_status "testing" .

####################
# credits and such #
####################
<http://moustaki.org/foaf.rdf#moustaki> a foaf:Person .

<http://kurtisrandom.com/foaf.rdf#kurtjx> a foaf:Person .
```

```
owl:Thing a owl:Class ;
        rdfs:isDefinedBy owl: .

foaf:Person a owl:Class, rdfs:Class ;
        rdfs:isDefinedBy foaf: .

dc:creator a owl:ObjectProperty ;
        rdfs:isDefinedBy dc: .

dc:title a owl:DatatypeProperty ;
        rdfs:isDefinedBy dc: .

vs:term_status a owl:AnnotationProperty ;
        rdfs:isDefinedBy vs: .
```

Bibliography

S. Abdallah, Y. Raimond, and M. Sandler. An ontology-based approach to information management for music analysis systems. In *120th Audio Engineering Society Convention*, 2006.

L. Adamic, R. Lukose, A. Puniyani, and B. Huberman. Search in power-law networks. *Physical review E*, 64(4):46135, 2001.

Y. Ahn, S. Han, H. Kwak, S. Moon, and H. Jeong. Analysis of topological characteristics of huge online social networking services. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 835–844, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-654-7. doi: http://doi.acm.org/10.1145/1242572.1242685. URL http://portal.acm.org/citation.cfm?id=1242685.

H. Alani and C. Brewster. Metrics for ranking ontologies. In *Proceedings of the 4th Int. Workshop on Evaluation of Ontologies for the Web*, 2006.

R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Rev. Mod. Phys.*, 74(1):47–97, Jan 2002. doi: 10.1103/RevModPhys.74.47.

R. Albert, H. Jeong, and A. Barabási. Error and attack tolerance of complex networks. *Nature*, 406(6794):378–382, 2000.

I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Lud
"ascher, and S. Mock. Kepler: An extensible system for design and execution of scientific workflows. In *Proceedings of the 16th International Conference on Scientific and Statistical Database Management (SSDBM04)*, volume 1099, pages 20–00, 2004.

J. I. Alvarez-Hamelin, L. Dall'Asta, A. Barrat, and A. Vespignani. k-core decomposition: a tool for the visualization of large scale networks. *CANADA*, page 41, 2006. URL http://www.citebase.org/abstract?id=oai:arXiv.org:cs/0504107.

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

C. Anderson and M. Andersson. *The long tail*. Hyperion New York, 2006.

R. Angles and C. Gutierrez. Survey of graph database models. *ACM Computing Surveys (CSUR)*, 40(1):1–39, 2008.

J.-J. Aucouturier and F. Pachet. A scale-free distribution of false positives for a large class of audio similarity measures. *Pattern Recogn.*, 41(1):272–284, 2008. ISSN 0031-3203. doi: http://dx.doi.org/10.1016/j.patcog.2007.04.012. URL http://portal.acm.org/citation.cfm?id=1285179.

S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. Dbpedia: A nucleus for a web of open data. *The Semantic Web*, pages 722–735, 2007.

C. Baccigalupo and E. Plaza. Poolcasting: A Social Web Radio Architecture for Group Customisation. 2007.

A. Bangor, P. Kortum, and J. Miller. An empirical evaluation of the system usability scale. *International Journal of Human-Computer Interaction*, 24 (6):574–594, 2008. ISSN 1044-7318.

A. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509, 1999.

A. Barabási and R. Crandall. Linked: The new science of networks. *American journal of Physics*, 71:409, 2003.

R. Barga, D. Fay, D. Guo, S. Newhouse, Y. Simmhan, and A. Szalay. Efficient scheduling of scientific workflows in a high performance computing cluster. In *Proceedings of the 6th international workshop on Challenges of large applications in distributed environments*, pages 63–68. ACM, 2008.

V. Batagelj and M. Zaversnik. Generalized cores, 2002. URL http://www.citebase.org/abstract?id=oai:arXiv.org:cs/0202039.

S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. McGuinness, P. Patel-Schneijder, and L. A. Stein. OWL Web Ontology Language Reference. Recommendation, World Wide Web Consortium (W3C), February10 2004. See http://www.w3.org/TR/owl-ref/.

D. Beckett. RDF/XML Syntax Specification (Revised). Recommendation, World Wide Web Consortium (W3C), 2004a. Internet: http://www.w3.org/TR/rdf-syntax/.

D. Beckett. Rdf/xml syntax specification (revised). http://www.w3.org/TR/REC-rdf-syntax/, 2004b.

D. Beckett and T. Berners-Lee. Turtle - terse RDF triple language, W3C team submission, 2008. URL http://www.w3.org/TeamSubmission/turtle/. See: http://www.w3.org/TeamSubmission/turtle/.

J. P. Bello. *Towards the Automated Analysis of Simple Polyphonic Music: A Knowledge-based Approach*. PhD thesis, Queen Mary University of London, 2003.

A. Berenzweig, B. Logan, D. P. W. Ellis, and B. P. W. Whitman. A large-scale evaluation of acoustic and subjective music-similarity measures. *Computer Music J.*, 28(2):63–76, 2004. ISSN 0148-9267. doi: http://dx.doi.org/10.1162/014892604323112257.

T. Berners-Lee. Linked data, July 2006. URL http://www.w3.org/DesignIssues/LinkedData.html.

T. Berners-Lee. Notation 3, 1998. URL http://www.w3.org/DesignIssues/Notation3. See http://www.w3.org/DesignIssues/Notation3.html.

T. Berners-Lee and D. Connolly. CWM-closed world machine. *Internet: http://www. w3. org/2000/10/swap/doc/cwm. html*, 2000.

T. Berners-Lee and M. Fischetti. *Weaving the Web: The original design and ultimate destiny of the World Wide Web by its inventor*. Harper San Francisco, 1999.

T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5), May 2001. URL http://www-sop.inria.fr/acacia/cours/essi2006/Scientific%20American_%20Feature%20Article_%20The%20Semantic%20Web_%20May%202001.pdf.

T. Berners-Lee, D. Connolly, L. Kagal, Y. Scharf, and J. Hendler. N3logic: A logical framework for the world wide web. *Theory and Practice of Logic Programming*, 2007. URL http://www.citebase.org/abstract?id=oai:arXiv.org:0711.1533.

C. Bizer, T. Heath, D. Ayers, and Y. Raimond. Interlinking open data on the web. In *Demonstrations Track, 4th European Semantic Web Conference*, 2007.

C. Bizer, T. Heath, K. Idehen, and T. Berners-Lee. Linked data on the web (ldow2008). In J. Huai, R. Chen, H.-W. Hon, Y. Liu, W.-Y. Ma, A. Tomkins, and X. Zhang, editors, *Proceedings of the 17th International Conference on World Wide Web (WWW)*, pages 1265–1266. ACM, 2008. ISBN 978-1-60558-085-2. URL http://dblp.uni-trier.de/db/conf/www/www2008.html#BizerHIB08.

C. Bizer, T. Heath, and T. Berners-Lee. Linked data-the story so far. *Int. J. Semantic Web Inf. Syst.*, 5(3):1–22, 2009.

H. Boley, S. Tabet, and G. Wagner. Design rationale of RuleML: A markup language for semantic web rules. In *International Semantic Web Working Symposium (SWWS)*, pages 381–402. Citeseer, 2001.

H. Boley, G. Hallmark, M. Kifer, A. Paschke, A. Polleres, and D. Reynolds. Rif core dialect. available at http://www.w3.org/TR/rif-core/, June 2010.

A. Bonner and M. Kifer. Concurrency and communication in transaction logic. In *Proceedings of the Joint International Conference and Symposium on Logic Programming*, 1996.

U. Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25(2):163–177, 2001.

C. Brewster, H. Alani, S. Dasmahapatra, and Y. Wilks. Data driven ontology evaluation. In *Proceedings of Language Resources and Evaluation*, pages 164–168, 2004.

D. Brickley and R. V. Guha. Rdf vocabulary description language 1.0: Rdf schema, February 2004. URL http://www.w3.org/TR/2004/REC-rdf-schema-20040210/. http://www.w3.org/TR/2004/REC-rdf-schema-20040210/.

S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117, 1998.

J. Brooke. SUS-A quick and dirty usability scale. *Usability evaluation in industry*, pages 189–194, 1996.

T. Buzan and B. Buzan. *The mind map book: how to use radiant thinking to maximize your brain's untapped potential*. Plume, 1996.

C. Cannam, M. O. Jewell, and C. Rhodes. Linked data and you: Bringing music research software into the semantic web. *Journal of New Music Research*, 2010.

P. Cano and M. Koppenberger. The emergence of complex network patterns in music artist networks. In *Proceedings of the 5th International Symposium on Music Information Retrieval (ISMIR04)*, 2004.

P. Cano, O. Celma, M. Koppenberger, and J. M. Buldu. The topology of music recommendation networks. arXiv.org:physics/0512266, 2005a. URL http://www.citebase.org/abstract?id=oai:arXiv.org:physics/0512266.

P. Cano, M. Koppenberger, and N. Wack. Content-based music audio recommendation. In *Proceedings of the 13th annual ACM international conference on Multimedia*, page 212. ACM, 2005b.

J. J. Carroll, C. Bizer, P. Hayes, and P. Stickler. Named graphs, provenance and trust. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 613–622, New York, NY, USA, 2005. ACM. ISBN 1-59593-046-9. doi: http://doi.acm.org/10.1145/1060745.1060835. URL http://portal.acm.org/citation.cfm?id=1060745.1060835.

M. Casey and M. Slaney. The importance of sequences in music similarity. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 5, pages 5–8, May 2006.

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

O. Celma. *Music Recommendation and Discovery in the Long Tail*. PhD thesis, Universitat Pompeu Fabra, Barcelona, Spain, 2008. URL http://mtg.upf.edu/~ocelma/PhD/doc/ocelma-thesis.pdf.

E. Chávez, G. Navarro, R. Baeza-Yates, and J. Marroquín. Searching in metric spaces. *ACM Computing Surveys (CSUR)*, 33(3):273–321, 2001.

L. F. Costa, F. A. Rodrigues, G. Travieso, and P. R. V. Boas. Characterization of complex networks: A survey of measurements. *Advances In Physics*, 56:167, 2007. URL doi:10.1080/00018730601170527.

E. Cox. Music on the internet: the internet as a new distribution medium for the music industry. http://www.edcox.net/about-me/research/music-on-the-internet/, 1998.

P. P. da Silva, D. L. McGuinness, and R. Fikes. A proof markup language for semantic web services. *Inf. Syst.*, 31(4):381–395, 2006. ISSN 0306-4379. doi: http://dx.doi.org/10.1016/j.is.2005.02.003. URL http://portal.acm.org/citation.cfm?id=1140602.

M. Davies. *Towards automatic rhythmic accompaniment*. PhD thesis, University of London, 2007.

I. Davis and R. Newman. Expression of core frbr concepts in rdf - working draft, 2005. URL http://vocab.org/frbr/core.

D. De Roure, C. Goble, and R. Stevens. The design and realisation of the Virtual Research Environment for social sharing of workflows. *Future Generation Computer Systems*, 25(5):561–567, 2009.

J. Downie, A. Ehmann, and D. Tcheng. Music-to-knowledge (M2K): a prototyping and evaluation environment for music information retrieval research. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, page 676. ACM, 2005.

H. Eisler and G. Ekman. A mechanism of subjective similarity. *Acta Psychologica*, 16:1–10, 1959.

A. Elberse. Should you invest in the long tail? *Harvard Business Review*, 86 (7/8):88–96, 2008.

P. Erdős and A. Rényi. On random graphs I. *Publ. Math. Debrecen*, 6: 290–297, 1959.

L. Euler. The königsberg bridge problem. *Commentarii academiae scientiarum Petropolitanae*, 8:128–140, 1741.

J. Euzenat and P. Valtchev. Similarity-based ontology alignment in OWL-lite. In *ECAI 2004: 16th European Conference on Artificial Intelligence, August 22-27, 2004, Valencia, Spain: including Prestigious Applicants [sic] of Intelligent Systems (PAIS 2004): proceedings*, page 333. Ios Pr Inc, 2004. ISBN 1586034529.

L. Feigenbaum, I. Herman, T. Hongsermeier, E. Neumann, and S. Stephens. The semantic web in action. *Scientific American Magazine*, 297(6):90–97, 2007.

R. Fielding. *Architectural styles and the design of network-based software architectures*. PhD thesis, Citeseer, 2000.

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 *Proc. of ICMC*, August 2008.

J. Foote, M. Cooper, and U. Nam. Audio retrieval by rhythmic similarity. In *Proceedings of the International Conference on Music Information Retrieval*, volume 3, pages 265–266. Citeseer, 2002.

L. C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, 1977.

T. Fruchterman and E. Reingold. Graph drawing by force-directed placement. *Software: Practice and Experience*, 21(11):1129–1164, 1991.

W. Garner. *The processing of information and structure*. L. Erlbaum Associates; distributed by Halsted Press, New York, 1974.

G. Giaquinto, C. Bledsoe, and B. McGuirk. Influence and Similarity between Contemporary Jazz Artists, plus Six Degrees of Kind of Blue. Master's thesis, University of Michigan, 2007.

M. Girvan and M. E. J. Newman. Community structure in social and biological networks. Working Papers 01-12-077, Santa Fe Institute, Dec. 2001. URL http://ideas.repec.org/p/wop/safiwp/01-12-077.html.

P. Gleiser and L. Danon. Community structure in jazz. *Advances in Complex Systems*, 6:565, 2003. URL doi:10.1142/S0219525903001067.

R. Goldstone. Alignment-based nonmonotonicities in similarity. *Journal of Experimental Psychology Learning Memory and Cognition*, 22:988–1001, 1996.

R. L. Goldstone and J. Y. Son. *The Cambridge Handbook of Thinking and Reasoning*, chapter Similarity, pages 13–36. Cambridge University Press, 2005.

F. Gouyon and S. Dixon. A review of automatic rhythm description systems. *Computer Music J.*, 29(1):34–54, 2005.

E. Heit and J. Rubinstein. Similarity and property effects in inductive reasoning. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 20:411–422, 1994.

M. Hepp. Possible ontologies: How reality constrains the development of relevant ontologies. *Internet Computing, IEEE*, 11(1):90–96, 2007. ISSN 1089-7801.

J. Hobbs and F. Pan. An ontology of time for the semantic web. *ACM Transactions on Asian Language Information Processing (TALIP)*, 3(1): 85, 2004.

K. Holyoak and K. Koh. Surface and structural similarity in analogical transfer. *Memory and cognition*, 15(4):332–340, 1987.

I. Horrocks, P. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. SWRL: A semantic web rule language combining OWL and RuleML. *W3C Member submission*, 21, 2004.

S. Imai. Pattern similarity and cognitive transformations* 1. *Acta Psychologica*, 41(6):433–447, 1977.

K. Jacobson. A multifaceted approach to music similarity. In *Proc. of ISMIR*, 2006.

K. Jacobson. Similarity ontology specification, May 2009. URL http://purl.org/ontology/musim.

K. Jacobson and M. Sandler. Musically meaningful or just noise, an analysis of on-line artist networks. In *Proc. of CMMR*, pages 306–314, 2008.

K. Jacobson, B. Fields, M. Casey, and M. Sandler. The effects of lossy audio encoding on genre classification tasks. In *Audio Engineering Society Convention 124*, 5 2008a. URL http://www.aes.org/e-lib/browse.cfm?elib=14547.

K. Jacobson, B. Fields, and M. Sandler. Using audio analysis and network structure to identify communities in on-line social networks of artists. *Proc. of ISMIR*, 2008b.

R. Jäschke, L. Marinho, A. Hotho, L. Schmidt-Thieme, and G. Stumme. Tag recommendations in folksonomies. *Knowledge Discovery in Databases: PKDD 2007*, pages 506–514, 2007.

T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information processing letters*, 31(12):7–15, 1989.

K. Kilkki. A practical model for analyzing long tails. *First Monday [Online]*, 12(5), May 2007.

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

R. Lambiotte and M. Ausloos. On the genre-fication of music: a percolation approach (long version). *The European Physical Journal B*, 50:183, 2006. URL doi:10.1140/epjb/e2006-00115-0.

R. Lambiotte and M. Ausloos. Uncovering collective listening habits and music genres in bipartite networks. *Physical Review E*, 72(6):66107, 2005.

O. Lassila and R. R. Swick, 1999. URL http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/.

P. Leach, M. Mealling, and R. Salz. A universally unique identifier (uuid) urn namespace. *RFC4122, July*, 2005.

S. H. Lee, P. J. Kim, and H. Jeong. Statistical properties of sampled networks. *Physical Review E*, 73:102–109, January 2006.

M. Levy and M. Sandler. Lightweight measures for timbral similarity of musical audio. In *Proceedings of the 1st ACM workshop on Audio and music computing multimedia*, page 36. ACM, 2006.

M. Levy, M. Sandler, and M. Casey. Extraction of high-level musical structure from audio data and its application to thumbnail generation. In *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, volume 5, pages V–V, 2006. ISBN 1-4244-0469-X. doi: 10.1109/ICASSP.2006.1661200. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1661200.

B. Logan. Mel frequency cepstral coefficients for music modeling. In *Proc. of ISMIR*, 2000.

B. Logan and A. Salomon. A music similarity function based on signal analysis. *Multimedia and Expo ICME*, pages 745–748, 2001.

A. Markman and D. Gentner. Structural alignment during similarity comparisons. *Cognitive Psychology*, 25:431–431, 1993.

W. Maurer. The dynamics of music distribution. http://www.chime.com/about/press/iris_online-9501.shtml, 1995.

L. Miller. Statement/statings. available at http://www.ilrt.bristol.ac.uk/discovery/2000/11/statements/, Nov 2000.

S. Mossa, M. Barthelemy, H. Eugene Stanley, and L. Nunes Amaral. Truncation of power law behavior in scale-free network models due to information filtering. *Physical Review Letters*, 88(13):138701, 2002.

M. E. J. Newman. Mixing patterns in networks. *Phys. Rev. E*, 67(2):026126, Feb 2003a. doi: 10.1103/PhysRevE.67.026126.

M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45:167, 2003b. URL http://www.citebase.org/abstract?id=oai:arXiv.org:cond-mat/0303516.

M. E. J. Newman and J. Park. Why social networks are different from other types of networks. *Phys. Rev. E*, 68(3):036122, Sep 2003. doi: 10.1103/PhysRevE.68.036122.

R. Nickerson. Binary-classification reaction time: A review of some studies of human information-processing capabilities. *Psychonomic Monograph Supplements*, 4:275–317, 1972.

K. Noland and M. Sandler. Signal processing parameters for tonality estimation. In *Proceedings of AES 122nd Convention*, Vienna, 2007.

L. Obrst, W. Ceusters, I. Mani, S. Ray, and B. Smith. The evaluation of ontologies. *Semantic Web*, pages 139–158, 2007.

T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Greenwood, T. Carver, M. Pocock, A. Wipat, and P. Li. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 2004.

I. S. G. on the Functional Requirements for Bibliographic Records. Functional requirements for bibliographic records: Final report, 1998. URL http://www.ifla.org/VII/s13/frbr/frbr1.htm.

F. Pachet. Knowledge management and musical metadata. *Encyclopedia of Knowledge Management*, 2005.

E. Pampalk. *Computational Models of Music Similarity and their Application in Music Information Retrival*. PhD thesis, Technischen Universität Wien, May 2006.

F. Pan. *Representing complex temporal phenomena for the semantic web and natural language*. PhD thesis, University of Southern California Los Angeles, CA, USA, 2007.

J. Park, O. Celma, M. Koppenberger, P. Cano, and J. M. Buldu. The social network of contemporary popular musicians. arXiv:physics/0609229, 2006. URL http://www.citebase.org/abstract?id=oai:arXiv.org:physics/0609229.

E. Prud'hommeaux and A. Seaborne. SPARQL query language for RDF, W3C recommendation, 2008. URL http://www.w3.org/TR/rdf-sparql-query/.

W. V. Quine. *Ontological relativity and other essays*. Columbia University Press, New York, NY, USA, 1969.

Y. Raimond. *A distributed music information system*. PhD thesis, Queen Mary University of London, 2009.

Y. Raimond and S. A. Abdallah. The event ontology. owl-dl ontology, 2006a. URL http://purl.org/NET/c4dm/event.owl.

Y. Raimond and S. A. Abdallah. The timeline ontology. owl-dl ontology, 2006b. URL http://purl.org/NET/c4dm/timeline.owl.

Y. Raimond, S. Abdallah, M. Sandler, and F. Giasson. The music ontology. Proceedings of the 8th ISMIR, 2007.

Y. Raimond, C. Sutton, and M. Sandler. Automatic interlinking of music datasets on the semantic web, 2008.

P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. GroupLens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186. ACM New York, NY, USA, 1994.

M. W. Richardson. Multidimensional psychophysics. *Psychological Bulletin*, 35:659–660, 1938.

G. Salton and M. McGill. *Introduction to modern information retrieval*. McGraw-Hill New York, 1983.

R. N. Shepard. Geometrical approximations to the structure of musical pitch. *Physchological Review*, 89:305–333, 1982.

B. Smith, W. Ceusters, B. Klagges, J. K
"ohler, A. Kumar, J. Lomax, C. Mungall, F. Neuhaus, A. Rector, and C. Rosse. Relations in biomedical ontologies. *Genome biology*, 6(5):R46, 2005.

C. H. Smith. The classical music navigator, 1993. URL http://www.wku.edu/~smithch/music/.

M. Sordo, O. Celma, M. Blech, and E. Guaus. The Quest for Musical Genres: Do the Experts and the Wisdom of Crowds Agree? In *Proceedings of the 9th International Conference on Music Information Retrieval*, page 255, 2008.

G. Stoilos, G. Stamou, and S. Kollias. A string metric for ontology alignment. *The Semantic Web–ISWC 2005*, pages 624–637, 2005.

M. Stumpf, C. Wiuf, and R. May. Subnets of scale-free networks are not scale-free: sampling properties of networks. *Proceedings of the National Academy of Sciences of the United States of America*, 102(12):4221, 2005.

I. Taylor. *Workflows for e-science: scientific workflows for grids*. Springer-Verlag New York Inc, 2007.

T. Teitelbaum, P. Balenzuela, P. Cano, and J. Buldú. Community structures and role detection in music networks. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 18:043105, 2008.

J. B. Tenenbaum. Learning the structure of similarity. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Advances in Neural Information Processing Systems 8*. MIT Press, Cambridge, MA, USA, 1996.

W. Torgerson. Multidimensional scaling of similarity. *Psychometrika*, 30(4): 379–393, 1965.

A. Tversky. Features of similarity. *Psychological review*, 84(4):327–352, 1977.

A. Tversky and I. Gati. Similarity, separability, and the triangle inequality. *Physchological Review*, 89:123–154, 1982.

G. Tzanetakis. MARSYAS Submissions to MIREX 2009. *Proceedings of 10th ISMIR*, 2009.

G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10:293– 302, 2002a. ISSN 1063-6676. doi: 10.1109/TSA.2002.800560. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1021072.

G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transactions on speech and audio processing*, 10(5):293–302, 2002b.

S. Ullman. *High-level vision*. MIT press Cambridge, MA, 1996.

D. Vrandečić. *Handbook on Ontologies*, chapter Ontology Evaluation, pages 293–313. Springer, 2009.

D. Watts and S. Strogatz. Collective dynamics of small-world networks. *Nature*, 393(6684):440–442, 1998.

B. Whitman, D. Roy, and B. Vercoe. Learning word meanings and descriptive parameter spaces from music. In *Proceedings of the HLT-NAACL 2003 workshop on Learning word meaning from non-linguistic data-Volume 6*, page 99. Association for Computational Linguistics, 2003.

B. A. Whitman and S. Lawrence. Inferring descriptions and similarity for music from community metadata. In *Proceedings of the 2002 International Computer Music Conference*, pages 591–598. Citeseer, 2002.

W. Wiener-Ehrlrich, W. Bart, and R. Millward. An analysis of generative representation systems. *Journal of Mathematical Psychology*, 21(3):21–246, 1980.