

HYPERGRAPH MODELS OF PLAYLIST DIALECTS

Brian McFee

Computer Science and Engineering
University of California, San Diego

Gert Lanckriet

Electrical and Computer Engineering
University of California, San Diego

ABSTRACT

Playlist generation is an important task in music information retrieval. While previous work has treated a playlist collection as an undifferentiated whole, we propose to build playlist models which are tuned to specific categories or *dialects* of playlists. Toward this end, we develop a general class of flexible and scalable playlist models based upon hypergraph random walks. To evaluate the proposed models, we present a large corpus of categorically annotated, user-generated playlists. Experimental results indicate that category-specific models can provide substantial improvements in accuracy over global playlist models.

1. INTRODUCTION

Playlist generation, the automated construction of sequences of songs, is a central component to online music delivery services. Because users tend to consume music sequentially in listening sessions, the quality of a playlist generation algorithm can significantly impact user satisfaction.

Recently, it has been proposed that playlist generation algorithms may be best viewed as probabilistic models of song sequences [11]. This viewpoint, borrowed from the statistical natural language processing literature, enables the automatic evaluation and optimization of a model by computing the likelihood of it generating examples of user-generated playlists. For this method to work, the practitioner must provide a large collection of example playlists, both for model evaluation and parameter optimization.

Of course, numerous subtleties and difficulties arise when working with user-generated playlist data. For example, the data is often noisy, and the author's intent may be obscure. In extreme cases, users may compose playlists by randomly selecting songs from their libraries. More generally, different playlists may have different intended uses (e.g., *road trip* or *party mix*), thematic elements (*break up* or *romantic*), or simply contain songs only of specific genres. While previous work treats the universe of user-generated playlists as a single *language*, building effective global models has proven to be difficult [11].

To better understand the structure of playlists, we advocate a more subtle approach. Rather than viewing naturally

occurring playlists as a single language, we propose to model playlists as a collection of *dialects*, each of which may exhibit its own particular structure. Toward this end, we develop dialect-specific playlist models, and evaluate on a large corpus of annotated, user-generated playlists.

The proposed approach raises several natural questions:

- Is it beneficial to individually model playlist dialects?
- Are some dialects easier to model than others?
- Which features are important for each dialect?

Answering these questions will hopefully provide valuable insight into the underlying mechanics of playlist generation.

1.1 Our contributions

In this work, our contributions are two-fold. First, we develop a flexible, scalable, and efficient class of generative playlist models based upon hypergraph random walks. Second, we present a new, large-scale, categorically annotated corpus of user-generated playlist data.

2. HYPERGRAPH RANDOM WALKS

Over the last decade, several researchers have proposed playlist generation algorithms based upon random walks [9, 11, 12].¹ Random walk playlist models consist of a weighted graph $G = (\mathcal{X}, E, w)$, where the vertices \mathcal{X} represent the library of songs, and the edges E and weights w encode pairwise affinities between songs. A playlist is then generated by following a random trajectory through the graph, where transitions $x_t \rightsquigarrow x_{t+1}$ are sampled according to the weights on edges incident to x_t .

Random walk models, while simple and efficient, carry certain practical limitations. It is often unclear how to define the weights, especially when multiple sources of pairwise affinity are available. Moreover, relying on pairwise interactions can severely limit the expressive power of these models (if each song has few neighbors), or scalability and precision (if each song has many neighbors).

To overcome these limitations, we propose a new class of playlist algorithms which allow for more flexible affinities between songs and sets of songs.

2.1 The user model

To motivate our playlist generation algorithm, we propose a simple model of user behavior. Rather than selecting songs

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page.

© 2012 International Society for Music Information Retrieval.

¹ There are many approaches beyond random walk models; see [5, chapter 2] for a survey of recent work.

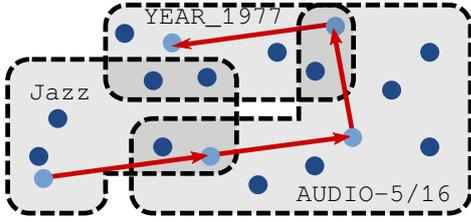


Figure 1. An example random walk on a song hypergraph: vertices represent songs, and edges are subsets of songs. Each transition $x_t \rightsquigarrow x_{t+1}$ must lie within an edge.

directly from the entire collection \mathcal{X} , we assume that the user first narrows her selection to a subset $e \subseteq \mathcal{X}$ (e.g., jazz songs), from which a song $x_0 \in e$ is chosen uniformly at random. For each subsequent transition $x_t \rightsquigarrow x_{t+1}$, the user selects a subset containing the current song x_t , and then selects x_{t+1} uniformly from that subset.

This user model is exactly characterized by a random walk on a *hypergraph*. Hypergraphs generalize undirected graphs by allowing an edge $e \in E$ to be an arbitrary subset of the vertices, rather than a pair (Figure 1). For example, a hypergraph edge may be as general as *jazz songs*, or as specific as *funk songs from 1977*. Edge weights can be used to encode the importance of a subset: for example, a model of jazz playlists would assign high weight to an edges containing jazz songs.

This model has several practically beneficial properties. First, it is efficient and scalable, in that the only information necessary to describe a song is its membership in the edge sets. Similarly, it naturally supports extension to new songs without having to significantly alter the model parameters (edge weights). Second, the model can easily integrate disparate feature sources, such as audio descriptors, lyrics, tags, *etc.*, as long as they can be encoded as subsets. Moreover, the model degrades gracefully if a song only has partial representation (e.g., audio but no lyrics or tags). Finally, the model is *transparent*, in that each transition can be explained to the user simply in terms of the underlying edge taken between songs. As we will see in Section 3, these edges often have natural semantic descriptions.

2.2 The playlist model

To formalize our model, let $H = (\mathcal{X}, E, w)$ denote a hypergraph over vertices (songs) \mathcal{X} , edges $E \subseteq 2^{\mathcal{X}}$, and non-negative weights $w \in \mathbb{R}_+^{|E|}$. We assume that the song library \mathcal{X} and edge set E are given, and our goal is to optimize the edge weights w . We denote by $x_t^e := \mathbb{1}[x_t \in e]$ the indicator that the song x_t is contained in the edge e .

Because the selection of the next song x_{t+1} depends only on the previous song x_t and edge weights w , the model is a first-order Markov process. The likelihood of a playlist $s = (x_0 \rightsquigarrow x_1 \rightsquigarrow \dots \rightsquigarrow x_T)$ thus factors into likelihood of the initial song, and each subsequent transition:

$$\mathbf{P}(x_0 \rightsquigarrow x_1 \rightsquigarrow \dots \rightsquigarrow x_T | w) = \mathbf{P}(x_0 | w) \prod_{t=0}^{T-1} \mathbf{P}(x_{t+1} | x_t, w).$$

Given the edge weights w , the distribution over the initial song x_0 can be characterized by marginalizing over edges:

$$\mathbf{P}(x_0 | w) := \sum_{e \in E} \mathbf{P}(x_0 | e) \mathbf{P}(e | w) = \sum_{e \in E} \frac{x_0^e w_e}{|e| \sum_{f \in E} w_f}.$$

Similarly, the probability of a transition $x_t \rightsquigarrow x_{t+1}$ is defined by marginalizing over edges incident to x_t :

$$\begin{aligned} \mathbf{P}(x_{t+1} | x_t, w) &:= \sum_{e \in E} \mathbf{P}(x_{t+1} | e, x_t) \mathbf{P}(e | x_t, w) \\ &= \sum_{e \in E} \frac{\mathbb{1}[x_{t+1} \neq x_t] \cdot x_{t+1}^e}{|e| - 1} \cdot \frac{x_t^e w_e}{\sum_{f \in E} x_t^f w_f}. \end{aligned}$$

Finally, to promote sparsity among the edge weights and resolve scale-invariance in the model, we assume an IID exponential prior on edge weights w_e with rate $\lambda > 0$:

$$\mathbf{P}(w_e) := \lambda \cdot \exp(-\lambda w_e) \cdot \mathbb{1}[w_e \in \mathbb{R}_+].$$

2.3 Learning the weights

Given a training sample of playlists $\mathcal{S} \subset \mathcal{X}^*$,² we would like to find the maximum a posteriori (MAP) estimate of w :

$$\begin{aligned} w &\leftarrow \operatorname{argmax}_{w \in \mathbb{R}_+^{|E|}} \log \mathbf{P}(w | \mathcal{S}) \\ &= \operatorname{argmax}_{w \in \mathbb{R}_+^{|E|}} \sum_{s \in \mathcal{S}} \log \mathbf{P}(s | w) + \sum_{e \in E} \log \mathbf{P}(w_e). \quad (1) \end{aligned}$$

The MAP objective (1) is not concave, and it is generally difficult to find a global optimum. Our implementation uses the L-BFGS-B algorithm [2] to solve for w , and converges quite rapidly to a stationary point. Training typically takes a matter of seconds, even for the large playlist collections and edge sets described in Section 3.

3. DATA COLLECTION

Previous work on playlist modeling used the Art of the Mix³ (AotM) collection of Ellis, et al. [4]. The existing AotM dataset was collected in 2002, and consists of roughly 29K playlists over 218K songs, provided as lists of plain-text song and artist names. In this work, we expand and enrich this dataset into a new collection, which we denote as AotM-2011.⁴ This section describes our data collection, pre-processing, and feature extraction methodology.

3.1 Playlists: Art of the Mix 2011

To expand the AotM playlist collection, we crawled the site for all playlists, starting from the first indexed playlist (1998-01-22) up to the most recent at the time of collection (2011-06-17), resulting in 101343 unique playlists. Each playlist contains not only track and artist names, but a timestamp and categorical label (e.g., *Road Trip* or *Reggae*).

² \mathcal{X}^* denotes the Kleene star operation.

³ <http://www.artofthemix.org>

⁴ <http://cosmal.ucsd.edu/cal/projects/aotm2011/>.

To effectively model the playlist data, the plain-text song and artist names must be resolved into a common namespace. Following previous work, we use the Million Song Dataset (MSD) as the underlying database [1, 11]. Rather than rely on the Echo Nest text-search API to resolve song identifiers, we instead implemented a full-text index of MSD song and artist names in Python with the Whoosh⁵ library. This allowed both high throughput and fine-grained control over accent-folding and spelling correction. Each (*artist*, *song*) pair in the raw playlist data was used as a query to the index, and resolved to the corresponding MSD song identifier (if one was found). In total, 98359 songs were matched to unique identifiers.

Because not every song in a raw playlist could be correctly resolved, each playlist was broken into contiguous segments of two or more matched song identifiers. Finally, playlist segments were grouped according to category. Table 1 lists each of the 25 most popular categories by size.

3.2 Edge features

To fully specify the playlist model, we must define the edges of the hypergraph. Because edges can be arbitrary subsets of songs, the model is able to seamlessly integrate disparate feature modalities. We use the following collection of edge features, which can be derived from MSD and its add-ons.

Audio To encode low-level acoustic similarity, we first mapped each song i to a vector $x_i \in \mathbb{R}^{222}$ using the optimized vector quantized Echo Nest Timbre (ENT) descriptors provided by [1, 11]. Audio descriptors were clustered via online k-means, and cluster assignments were used to produce k disjoint subsets. Repeating this for $k \in \{16, 64, 256\}$ provided multiple overlapping edges of varying degrees of granularity. All 98K songs receive audio representations.

Collaborative filter To capture high-level similarities due to user listening patterns, we construct edges from the taste profile data used in the MSD Challenge [10]. We used the Bayesian Personalized Ranking (BPR) algorithm [6, 13] to factor the users-by-songs (1M-by-380K) feedback matrix into latent feature vectors $x_i \in \mathbb{R}^{32}$. The BPR regularization parameters were set to $\lambda_1 = \lambda_2 = 10^{-4}$. Edges were constructed by cluster assignments following the procedure described above for audio features. 62272 songs (63%) coincide with the taste profile data.

Era The era in which songs are released can play an important role in playlist composition [3, 8]. To model this, we use the MSD meta-data to represent each song by its year and half-overlapping decades. For example, the song *Parliament - Flash Light* maps to edges YEAR-1977, DECADE-1970 and DECADE-1975. 77884 songs (79%) were mapped to era descriptors.

Familiarity Previous studies have noted the importance of song- or artist-familiarity when composing playlists [3,

11]. We used the artist familiarity data provided with MSD, which maps each song to the range $[0, 1]$ (0 being unfamiliar, 1 being very familiar). Edges were constructed by estimating the 25th and 75th percentiles of familiarity, and mapping each song to LOW, MEDIUM, or HIGH familiarity.

Lyrics Previous studies have shown the importance of lyrics in playlist composition [8]. To compute lyrical similarity, we applied online latent Dirichlet allocation (LDA) [7] with $k = 32$ to the musixmatch lyrics database.⁶ We then constructed three sets of 32 edges (one edge per topic): the first matches each song to its most probable topic, the second matches each song to its top three topics, and the third set to its top five topics. 53351 songs (56%) were found in the musixmatch data.

Social tags Previous work incorporated semantic information by using the total similarity between bag-of-tags vectors of songs to determine similarity [11]. Here, we take a more flexible approach, and model each tag separately. Using the Last.fm⁷ tags for MSD, we match each song to its top-10 most frequent tags. Each tag induces an edge (the songs assigned to that tag).⁸ 80396 songs (82%) matched to tag edges.

Uniform shuffle Because the features described above cannot model all possible transitions, we include a uniform edge that contains all songs. A transition through the uniform edge can be interpreted as a random restart of the playlist. The uniform shuffle also provides a standard baseline for comparison.

Feature conjunctions Some of the features described above may be quite weak individually, but when combined, become highly descriptive. For example, the tag `rock` and era `YEAR-1955` are both vague, but the conjunction of these two descriptors — `rock-&-YEAR-1955` — retains semantic interpretability, and is much more precise. We therefore augment the above collection of edges with all pair-wise intersections of features. Note that this induces general cross-modal feature conjunctions, such as `Lyrics topic #4-&-Audio cluster #17`, resulting in an extremely rich set of song descriptors.

4. EXPERIMENTS

To evaluate the proposed method, we randomly partitioned each of the top-25 categories listed in Table 1 into ten 75/25 train/test splits. For each split, the train (test) sets are collected across categories to form a global train (test) set *ALL*, which is used to train a global model. After fitting a

⁵ <http://packages.python.org/Whoosh/>

⁶ <http://labrosa.ee.columbia.edu/millionsong/musixmatch>

⁷ <http://last.fm/>

⁸ A similar tag-hypergraph model was proposed by Wang, et al. [14].

Category	Playlists	Segments	Songs	Category	Playlists	Segments	Songs
<i>Mixed</i>	41798	101163	64766	<i>Sleep</i>	675	1487	2957
<i>Theme</i>	12813	31609	35862	<i>Electronic Music</i>	611	1131	2290
<i>Rock-Pop</i>	4935	13661	20364	<i>Dance-House</i>	526	1117	2375
<i>Alternating DJ</i>	4334	10493	18083	<i>Rhythm and Blues</i>	432	1109	2255
<i>Indie</i>	4528	10333	13678	<i>Country</i>	398	908	1756
<i>Single Artist</i>	3717	9044	17715	<i>Cover</i>	447	833	1384
<i>Romantic</i>	2523	6269	8873	<i>Hardcore</i>	268	633	1602
<i>Road Trip</i>	1846	4817	8935	<i>Rock</i>	215	565	1866
<i>Punk</i>	1167	3139	4936	<i>Jazz</i>	295	512	1089
<i>Depression</i>	1128	2625	4794	<i>Folk</i>	241	463	1137
<i>Break Up</i>	1031	2512	4692	<i>Reggae</i>	183	403	831
<i>Narrative</i>	964	2328	5475	<i>Blues</i>	165	373	892
<i>Hip Hop</i>	1070	1958	2505	<i>Top-25</i>	86310	209485	97411

Table 1. The distribution of the top 25 playlist categories in AotM-2011. Each playlist consists of one or more segments of at least two contiguous MSD songs. 948 songs do not appear within the top 25 categories, but are included in the model.

Feature	# Edges	Feature	# Edges
Audio	204	Collaborative filter	93
Era	56	Familiarity	3
Lyrics	82	Tags	201
Uniform	1	All features	640
		Feature conjunctions	6390

Table 2. Summary of edges after pruning.

model to each training set, we compute the average (length-normalized) log-likelihood of the test set \mathcal{S}' :

$$\mathcal{L}(\mathcal{S}' | w) := \frac{1}{|\mathcal{S}'|} \sum_{s \in \mathcal{S}'} \frac{1}{|s|} \log \mathbf{P}(s | w).$$

For comparison purposes, we report performance in terms of the relative gain over the uniform shuffle model w_u (all weight assigned to the uniform edge):

$$G(w) := 1 - \frac{\mathcal{L}(\mathcal{S}' | w)}{\mathcal{L}(\mathcal{S}' | w_u)}.$$

To simplify the model and reduce over-fitting effects, we pruned all edges containing fewer than 384 (98359/256) songs. Similarly, we pruned redundant conjunction edges that overlapped by more than 50% with either of their constituent edges. Table 2 lists the number of edges retained after pruning. On average, each song maps to 76.46 ± 57.79 edges, with a maximum of 218. In all experiments, we fix the prior parameter $\lambda = 1$.

4.1 Experiment 1: Does dialect matter?

In the first set of experiments, we compare the global model to category-specific models. Figure 2 illustrates the relative gain over uniform across all categories for four different model configurations: tags, tags with pairwise conjunctions, all features, and all features with conjunctions.

Several interesting trends can be observed from Figure 2. First, in all but two cases — *Narrative* and *Rock* under the all features with conjunctions model — category-specific models perform at least as well as the global model, and are often substantially better. As should be expected, the effect is most pronounced for genre-specific categories that naturally align with semantic tags (e.g., *Hip Hop* or *Punk*).

Note that the larger categories overlap more with *ALL*, leaving less room for improvement over the global model.

Not surprisingly, the *Mixed* category appears to be difficult to model with similarity-based features. The fact that it is the single largest category (Table 1) may explain some of the difficulties observed in previous studies when using a global model [11]. Similarly, several other categories are quite broad (*Theme*, *Narrative*, *Rock*), or may be inherently difficult (*Alternating DJ*, *Mixed*).

Also of note are differences across model configurations. Feature conjunctions generally provide a modest improvement, both in the global and category-specific models. Due to the large parameter space, some over-fitting effects can be observed in the smallest categories (*Folk*, *Reggae*, *Blues*). Interestingly, several categories benefit substantially from the inclusion of all features compared to only tags (e.g., *Hip Hop*, *Punk*, *Jazz*).

4.2 Experiment 2: Do transitions matter?

Given the flexibility of the model, it is natural to question the importance of modeling playlist continuity: could a model which ignores transition effects perform as well as the random walk model? To test this, we split each playlist $s = (x_0 \rightsquigarrow x_1 \rightsquigarrow \dots \rightsquigarrow x_T)$ into singletons $s_0 = (x_0)$, \dots , $s_T = (x_T)$. With this modified corpus, the model treats each song in a playlist as an independent draw from the initial song distribution $\mathbf{P}(x_0 | w)$. Consequently, a model trained on this corpus can fit global trends across playlists within a category, but cannot enforce local continuity.

Figure 3 illustrates the relative gain for each category under the stationary distribution with all features and conjunctions. The results are qualitatively similar for alternate model configurations. Compared to Figure 2 (bottom-right), the results are substantially worse for most categories. In many cases, the stationary model performs worse than the uniform shuffle. This reflects the importance of transition effects when modeling playlists, even when the corpus is confined to genre-specific categories.

4.3 Experiment 3: Which features matter?

As illustrated in Figure 2, certain categories seem to benefit substantially from the inclusion of non-tag features. To investigate this effect, Figure 4 illustrates the aggregated weight for each feature type under each of the category models. Note that weight is aggregated across feature con-

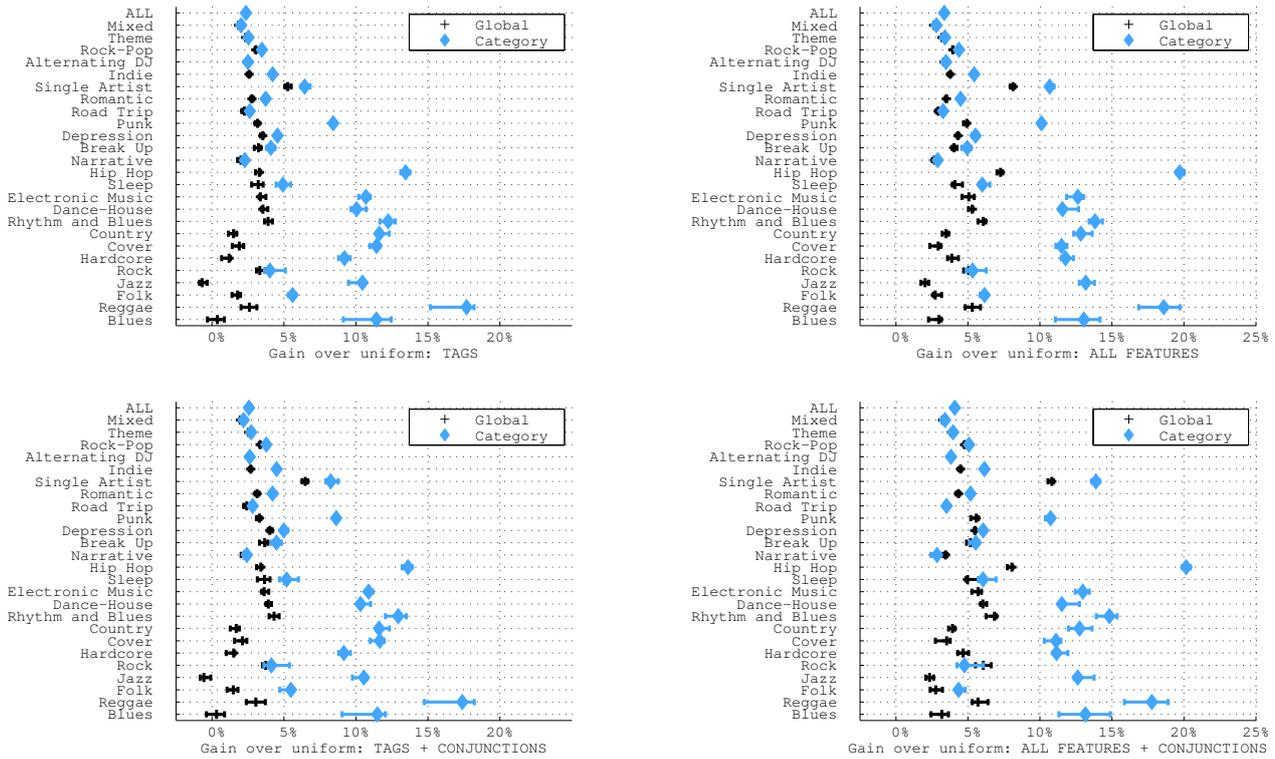


Figure 2. The median gain in log-likelihood over the uniform shuffle model, aggregated over ten random splits of the data. Error bars span the 0.25–0.75 quantiles. Category-specific models generally outperform global models.

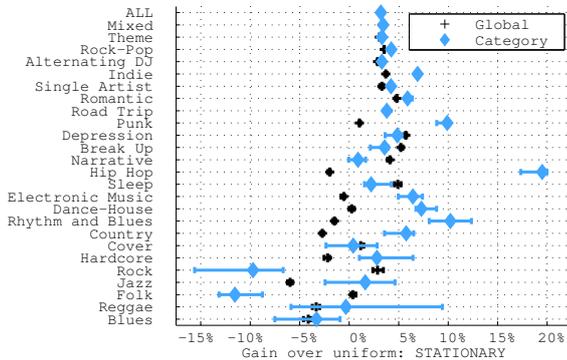


Figure 3. Log-likelihood gain over uniform with the stationary model (all features and conjunctions). Ignoring temporal structure significantly degrades performance.

junctions, so the weight for edge `DECADE_1955-&-Rock` counts both for *Era* and *Tag*.

Tags receive the most weight (64% on average) across all categories. Audio features appear to be most useful in *Hip Hop*, *Jazz* and *Blues* (43%–44%, compared to 26% average). This is not surprising, given that these styles feature relatively distinctive instrumentation and production qualities. Lyrical features receive the most weight in categories with salient lyrical content (*Folk*, *Cover*, *Narrative*, *Hardcore*, *Break Up*) and low weight in categories with little or highly variable lyrical content (*Electronic Music*, *Dance-House*, *Jazz*). Era and familiarity receive moderate weight (on aver-

age, 22% and 15% respectively), but the majority (20% and 14%) is due to conjunctions.

4.4 Example playlists

Table 3 illustrates samples drawn from category-specific feature conjunction models. For generative purposes, the uniform edge was removed after training. The generated playlists demonstrate both consistency within a single playlist and variety across playlists. Each transition in the playlist is explained by the corresponding (incoming) edge, which provides transparency to the user: for example, *Cole Porter - You're the Top* follows *Django Rheinhardt - Brazil* because both songs belong to the conjunction edge `AUDIO-3/16-&-jazz`, and share both high- and low-level similarity.

5. CONCLUSION

We have demonstrated that playlist model performance can be improved by treating specific categories of playlists individually. While the simple models proposed here work well in some situations, they are far from complete, and suggest many directions for future work. The first-order Markov assumption is clearly a simplification, given that users often create playlists with long-term interactions and global thematic properties. Similarly, the uniform distribution over songs within an edge set allows for an efficient and scalable implementation, but allowing non-uniform distributions could also be an avenue for future improvement.

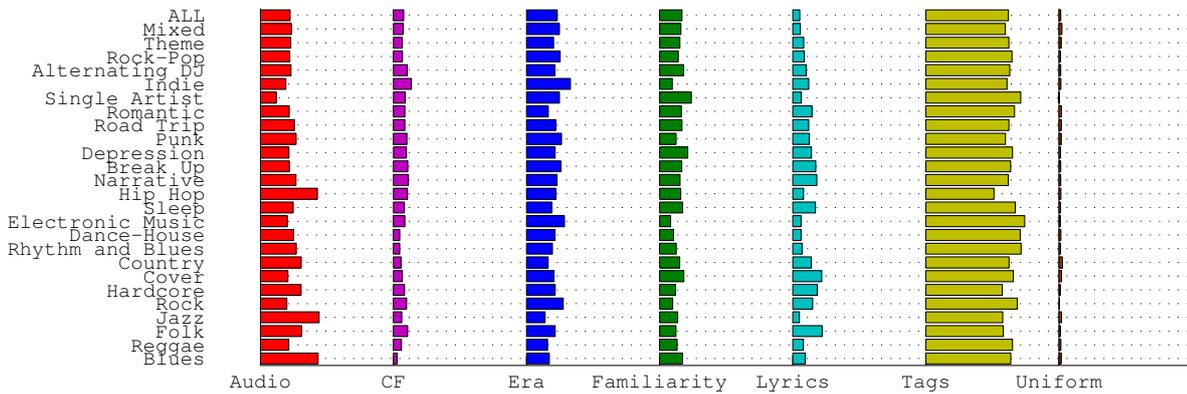


Figure 4. Distribution of learned edge weights for each playlist category. Weight is aggregated across feature conjunctions.

Category	Edge	Playlist
Hip Hop	AUDIO-149/256	<i>Eminem - The Conspiracy (Freestyle)</i>
	AUDIO-149/256	<i>Busta Rhymes - Bounce</i>
	DECADE-2000-&-rap	<i>Lil' Kim (Featuring Sisqo) - How Many Licks?</i>
	old school DECADE_1985-&-Hip-Hop AUDIO-12/16	<i>A Tribe Called Quest - Butter</i> <i>Beastie Boys - Get It Together</i> <i>Big Daddy Kane - Raw [Edit]</i>
Electronic Music	AUDIO-11/16-&-downtempo DECADE_1990-&-trip-hop	<i>Everything But The Girl - Blame</i> <i>Massive Attack - Spying Glass</i>
	AUDIO-11/16-&-electronica	<i>Björk - Hunter</i>
	DECADE_2000-&-AUDIO-23/64 electronica-&-experimental electronica-&-experimental	<i>Four Tet - First Thing</i> <i>Squarepusher - Port Rhombus</i> <i>The Chemical Brothers - Left Right</i>
	70s-&-soul AUDIO-14/16-&-funk DECADE_1965-&-soul	<i>Lyn Collins - Think</i> <i>Isaac Hayes - No Name Bar</i> <i>Michael Jackson - My Girl</i>
Rhythm and Blues	AUDIO-6/16-&-soul FAMILIARITY_MED-&-60s soul-&-oldies	<i>The Platters - Red Sails In The Sunset</i> <i>The Impressions - People Get Ready</i> <i>James & Bobby Purify - I'm Your Puppet</i>
	AUDIO-14/16-&-jazz	<i>Peter Dinklage - St Louis Blues</i> <i>Tony Bennett - The Very Thought Of You</i> <i>Louis Prima - Pennies From Heaven</i>
	jazz vocal jazz	<i>Django Reinhardt - Brazil</i> <i>Cole Porter - You're The Top</i> <i>Doris Day - My Blue Heaven</i>
Jazz	jazz-&-instrumental AUDIO-3/16-&-jazz	<i>Django Reinhardt - Brazil</i> <i>Cole Porter - You're The Top</i> <i>Doris Day - My Blue Heaven</i>
	jazz	<i>Doris Day - My Blue Heaven</i>

Table 3. Example playlists generated by various dialect models. *Edge* denotes the incoming edge to the corresponding song, which for transitions, is shared by the previous song. Feature conjunctions are indicated by X-&-Y.

6. ACKNOWLEDGMENTS

The authors acknowledge support from Qualcomm, Inc., eHarmony, Inc., Yahoo! Inc., Google, Inc., and NSF Grants CCF-0830535, IIS-1054960, and EIA-0303622.

7. REFERENCES

- [1] T. Bertin-Mahieux, D.P.W. Ellis, B. Whitman, and P. Lamere. The million song dataset. In *ISMIR*, 2011.
- [2] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM J. Sci. Comput.*, 16(5):1190–1208, Sep. 1995.
- [3] S.J. Cunningham, D. Bainbridge, and A. Falconer. “More of an art than a science”: Supporting the creation of playlists and mixes. In *ISMIR*, 2006.
- [4] D. Ellis, B. Whitman, A. Berenzweig, and S. Lawrence. The quest for ground truth in musical artist similarity. In *ISMIR*, 2002.
- [5] B. Fields. *Contextualize Your Listening: The Playlist as Recommendation Engine*. PhD thesis, Goldsmiths, University of London, April 2011.
- [6] Z. Gantner, S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. MyMediaLite: A free recommender system library. In *RecSys*, 2011.
- [7] M. Hoffman, D. Blei, and F. Bach. Online learning for latent dirichlet allocation. In *NIPS*. 2010.
- [8] J.H. Lee. How similar is too similar?: Exploring users’ perceptions of similarity in playlist evaluation. In *ISMIR*, 2011.
- [9] B. Logan. Content-based playlist generation: exploratory experiments. In *ISMIR*, 2002.
- [10] B. McFee, T. Bertin-Mahieux, D.P.W. Ellis, and G.R.G. Lanckriet. The million song dataset challenge. In *AdMIRe*, 2012.
- [11] B. McFee and G. R. G. Lanckriet. The natural language of playlists. In *ISMIR*, 2011.
- [12] R. Ragno, C. J. C. Burges, and C. Herley. Inferring similarity between music objects with application to playlist generation. In *7th ACM SIGMM international workshop on Multimedia information retrieval*, 2005.
- [13] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*, 2009.
- [14] F. Wang, X. Wang, B. Shao, T. Li, and M. Ogihara. Tag integrated multi-label music style classification with hypergraph. In *ISMIR*, 2009.