Embedding Songs and Tags for Playlist Prediction

Joshua L. Moore JLMO@CS.CORNELL.EDU
Shuo Chen SHUOCHEN@CS.CORNELL.EDU
Thorsten Joachims TJ@CS.CORNELL.EDU
Cornell University, Ithaca, NY, USA

Douglas Turnbull DTURNBULL@ITHACA.EDU
Ithaca College, Ithaca, NY, USA

1. Introduction

Automatic playlist generation can be a useful tool to navigate the myriad choices available to users in music services today. Here, we present our recent work on explicitly modeling playlists without requiring external similarity measures. Our *Logistic Markov Embedding* is trained directly on historical playlist data and can unify songs and (when available) social tags in a Euclidean space. The resulting space can be used to generate playlists, perform tag-based retrieval tasks, or to visualize songs and tags.

Related commerical approaches include Pandora and Apple iTunes Genius, which rely on analysis by human experts and collaborative filtering respectively to generate recommendations. However, the method of ordering that these methods employ is unknown, and it is not known how well they fare in rigorous evaluations. Other works have learned transition quality discriminatively (Maillet et al., 2009) or through generative models using song features (McFee & Lanckriet, 2011). However, both methods use acoustic and/or social tag similarity, whereas our method requires no such information about songs. Furthermore, previous work has been done in embedding songs into a similarity-based music space (e.g., (Weston et al., 2011)) for retrieval purposes. Our method differs from these since it explicitly models the sequential nature of playlists.

2. Probabilistic Embedding Model

We use a first order Markov assumption, reducing the probability of a playlist $p = (p^{[1]}, ..., p^{[k_p]})$ to $\prod_{i=1}^{k_p} \Pr(p^{[i]}|p^{[i-1]})$. We assume that each tag t and each song $s_i \in \mathcal{S}$ (with possibly empty tag set $T(s_i)$) can be embedded into a latent space of dimension d (a parameter of the model) by an embedding function $X(\cdot)$. The tag embeddings are unconstrained, whereas each song is assumed to be related to its tags via the equation $X(s_i) = \sum_{t \in T(s_i)} \frac{X(t)}{|T(s_i)|} + \bar{X}(s_i)$, where $\bar{X}(s_i)$ is unconstrained except for regularization. Finally, we model $\Pr(p^{[i]}|p^{[i-1]})$ as $\frac{e^{-||X(p^{[i]})-X(p^{[i-1]})||_2^2}}{\sum_{s \in \mathcal{S}} e^{-||X(s)-X(p^{[i-1]})||_2^2}}$, and we optimize the log-likelihood over all playlists in the data set via stochastic gradient descent to find the most probable embedding for the data. A regularizer can be added to the objective in order to penalize $\lambda \sum_{s \in \mathcal{S}} ||\bar{X}(s)||_2^2$ (where λ is a free parameter) to encourage the model to explain songs using their tags.

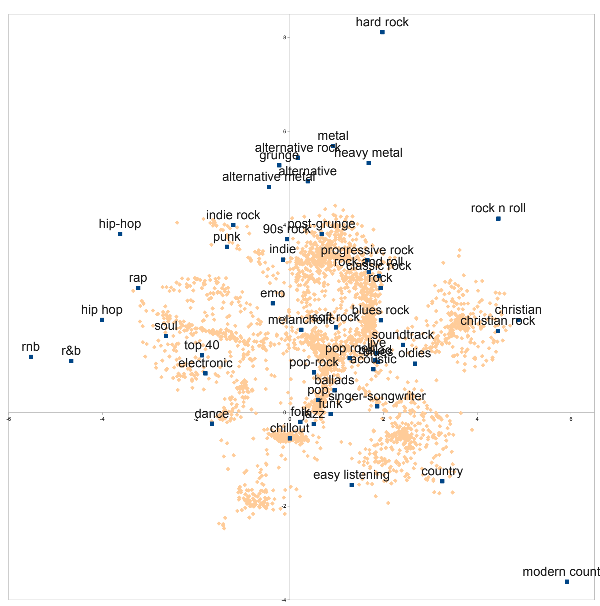

Figure 1. 2D embedding from LME with tags. The top 50 genre tags are labeled; lighter points represent songs.

In *5th International Workshop on Machine Learning and Music*, Edinburgh, Scotland, UK, 2012.

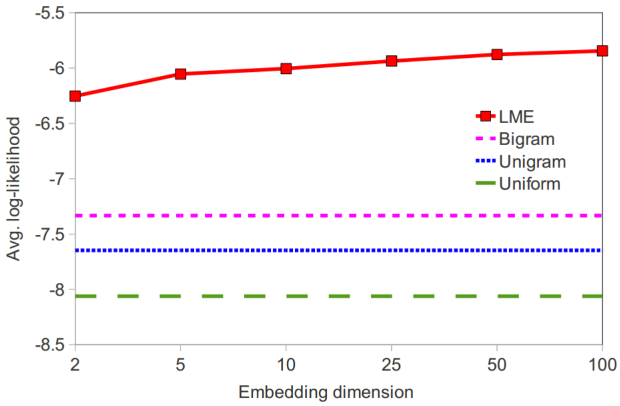

Figure 2. Log-likelihood on test set for LME and baselines.

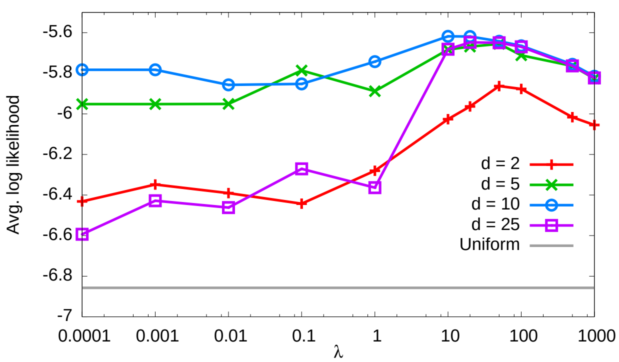

Figure 3. Log-likelihood of predicting transitions for new songs for different dimension d and regularization coefficient λ.

3. Experiments

We crawled radio playlists from Yes.com, obtaining 75,262 songs and 2,840,553 pairwise transitions. We then pruned the data, rejecting songs occurring fewer than 20 times, yielding 3,168 songs and 1,325,710 transitions. We use a train/test split of roughly 10%/90% over playlists. We crawled tag information for each song from Last.fm, keeping the 250 tags occurring most frequently in our songs. We present the results of four experiments. In Figure 1, a visualization of embeddings of genre tags and songs is presented. In Figure 2, we compare the modeling power of our method (using log-likelihood of generating test playlists as the objective) to three transition probability baselines: uniform (over songs), unigram (transition to a song is proportional to the song's frequency in the data), and smoothed bigrams (probability of a transition is proportional to its frequency in the data; Witten-Bell smoothing is used to estimate probability of unseen transitions). Figure 3 presents results of testing on transitions from seen songs to unseen songs using our tag model with varying regularization strength.

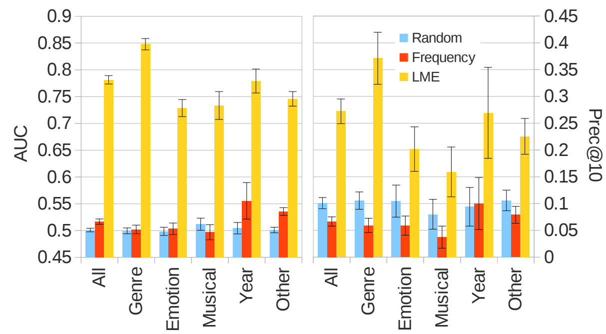

Figure 4. Average AUC (left) and precision at 10 (right) across tag categories for random and frequency baselines and LME. Error bars indicate +/- 1 standard error.

Finally, Figure 4 shows the AUC and precision at 10 of our method and two baselines on a retrieval task. Here, held out test songs (which have at least one tag) have their tags removed and an embedding is learned using all songs and playlists. We query each tag for related songs from the test set, retrieving songs ranked by distance to the query tag in the learned embedding space. If the original tags for a song included an exact lexicographic match for the query tag (i.e. "hip hop" is not the same as "hip-hop"), the song is considered relevant. As baselines we use a random ranking and a ranking by the number of times a song occurs in the data set. We further break down the tags into categories: genre, emotion, musical (non-genre music descriptors like "major key tonality" or "male vocals"), years/decades, and other (including opinion tags like "awesome").

4. Conclusions

We presented an embedding method based on a probabilistic model of playlist generation. The method yields a unified metric between songs and tags, and we demonstrated its effectiveness for playlist generation, generalization to unseen songs, and a song retrieval task.

References

Maillet, F., Eck, D., Desjardins, G., and Lamere, P. Steerable playlist generation by learning song similarity from radio station playlists. In *ISMIR*, 2009.

McFee, B. and Lanckriet, G. R. G. The natural language of playlists. In *ISMIR*, 2011.

Weston, J., Bengio, S., and Hamel, P. Multi-tasking with joint semantic spaces for large-scale music annotation and retrieval. *Journal of New Music Research*, 2011.