Blending Two Playlist Generation Algorithms

UNION COLLEGE

James E. Curbow

Matthew Anderson, Advisor

CS

Introduction

We blend two existing automatic playlist generation algorithms. One algorithm is built to transition between a start song and an end song (Start-End) [1]. The other focuses on the smoothness of transitions by inferring song similarity based on adjacent occurrences in expertly authored streams (EAS) [2]. First we seek to establish the effectiveness of the Start-End algorithm using the EAS algorithm to determine transition smoothness, then we implement improvements to optimize the Start-End algorithm.

Expertly Authored Streams

The idea: songs that appear adjacent in an expertly curated playlist have similarities — an expert (such as a DJ) probably wouldn't follow up Beethoven's 5th Symphony with Justin Bieber's latest single. The EAS algorithm exploits these similarities.

The EAS algorithm is implemented using a graph data structure, in which each song is represented by a node. Each node maintains a dictionary that keeps track of neighboring nodes. The weight of an edge between two songs equals the number of playlist adjacencies between those songs. From edge weight we can derive similarity.

Playlist

1	Song A
2	Song B
3	Song C
4	Song B
5	Song A
6	Song C
7	Song D

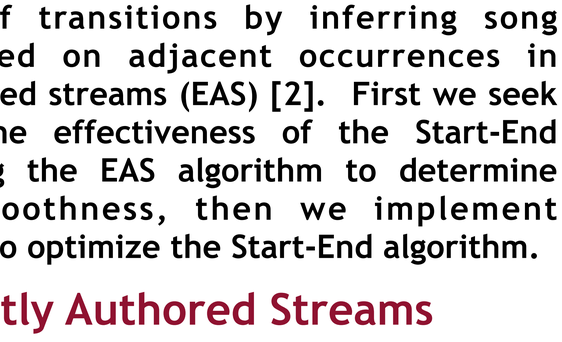
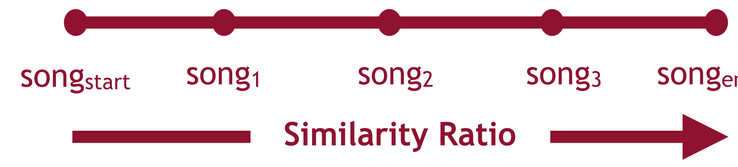


Figure 1. The EAS Algorithm: Edge weights equal the number of adjacent occurrences between songs in the playlist

The Start & End Song Algorithm

The Start-End algorithm picks and orders songs based on increasing similarity ratio. The similarity ratio is a song's similarity to the end song divided by its similarity to the start song.



Unbiased Random Walk Algorithm

The Unbiased Random Walk (URW) algorithm performs a random walk on the graph produced by the EAS algorithm. It's starting node is the node corresponding to the start song, and the target node is the node corresponding to the end song. The algorithm finds a random path to the end song.

We can make the algorithm greedy by choosing the nodes that it adds to the path in a slightly less random fashion—by basing the choices off of similarity we aim to create smooth transitions between songs in the playlist. We implemented a similar algorithm, the Biased Random Walk, that chooses songs based on probability as well.

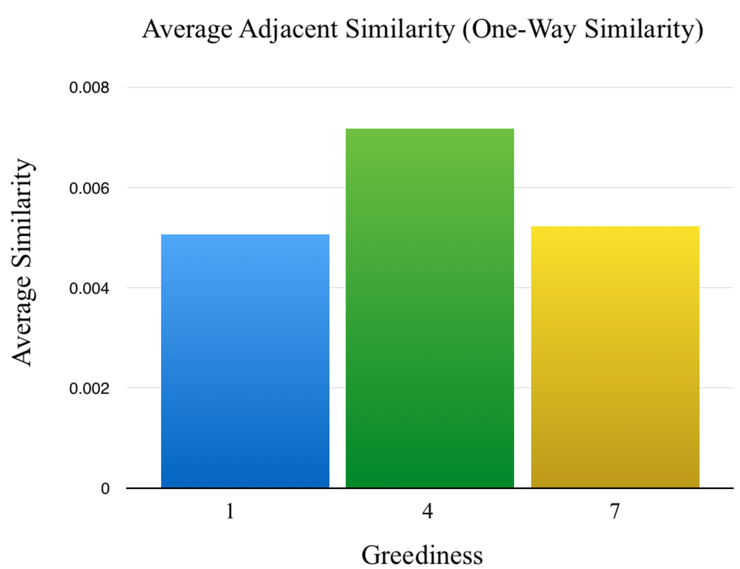


Figure 2: Average Adjacent Similarities

Results & Conclusion

A flaw in the Start-End algorithm prevented us from collecting results. Every song in the playlist must have a non-zero similarity to both the start song and then end song. This is a very restrictive factor because this limits the number of songs we can choose from. Our data set, while large enough for the URW algorithm, was not large enough to work with the Start-End algorithm.

Our results for the URW algorithm indicate that it performs better than the Start-End algorithm. We used adjacent similarity (the similarity between each song and its successor) as an indicator for playlist smoothness. The results shown are the use our one-way similarity calculation which is symmetric. The bar graph in Figure 2 suggests that the URW algorithm has a greediness "sweet-spot". Once the algorithm gets too greedy, it struggles to make a smooth playlist. The boxes in the matrices in Figure 3 are shaded based on similarity. The dark band indicates high similarity.

Our goal was to determine the effectiveness of the Start-End algorithm and to improve it. We established that the algorithm is not effective and that the URW is a significant improvement.

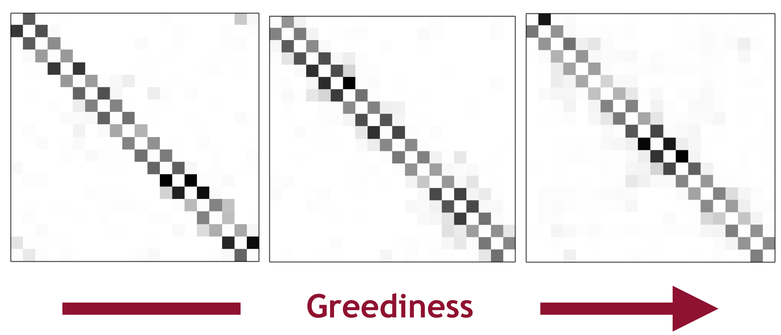


Figure 3: Similarity Matrices

References

[1]A. Flexer, D. Schnitzer, M. Gasser, and G. Widmer. Playlist generation using start and end songs. In *ISMIR*, pages 173-178, 2008.

[2]R. Ragno, C. Burges, and C. Herley. Inferring similarity between music objects with application to playlist generation. In *Proceedings of the 7th ACM SIGMM international workshop on MIR*, pages 73-80. ACM, 2005.